MICROHOBBY

AMSTRAD Especial

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO I N.º 2

475 ptas.

SOFTWARE INTEGRADO 3 PROGRAMAS EN UNO

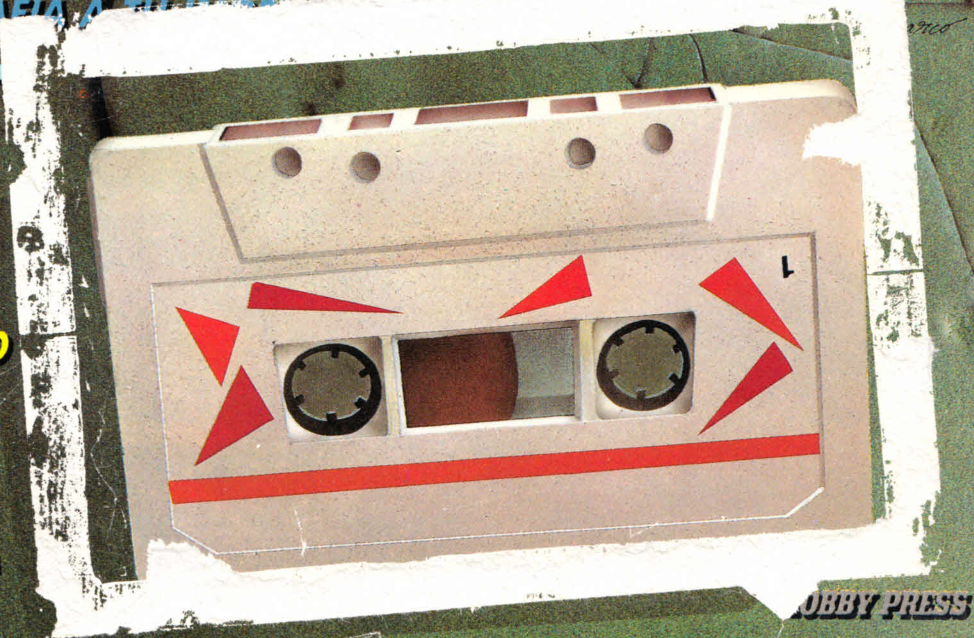
TABLETA GRAFICA GRAFPAD II: EL ARTE POR ORDENADOR A TU ALCANCE

TE OFRECEMOS UN LENGUAJE DE PROGRAMACION LISP COMPLETO EN CINTA DE CASSETTE

NUESTRO PROGRAMA CREADOR DE CRUCIGRAMAS DESAFIA A

IMPRESORAS: COMO HACER LA MEJOR ELECCION

ATENCION A NUESTRO FABULOSO CONCURSO: PUEDES GANAR UN CPC-6128 CON SOLO CARGAR LA CINTA



HOBBY PRESS

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
José M.ª Díaz
Redactor Jefe
Juan José Martínez
Diseño gráfico
Fernando Chaumel
Colaboradores
Eduardo Ruiz
Javier Barceló
David Sopuerta
Robert Chatwin
Francisco Portalo
Pedro Sudón
Miguel Sepúlveda
Francisco Martín
Jesús Alonso
Pedro S. Pérez
Amalio Gómez
Secretaria Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Portada
M. Barco
Ilustradores
J. Igual, J. Pons, F. L. Frontán,
J. Septien, Pejo, J. J. Mora

Edita
HOBBY PRESS, S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Producción
Carlos Peropadre

Marketing
Marta García
Jefe de Publicidad
Concha Gutiérrez
Publicidad Barcelona
José Galán Cortés
Tel: (93) 303 10 22/313 71 62

Secretaria de Dirección
Marisa Cogorro

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

Redacción, Administración y Publicidad
Ctra. de Irún km 12,400
(Fuencarral) 28049 Madrid
Teléfonos: Suscrip.: 734 65 00
Redacción: 734 70 12

Dto. Circulación
Paulino Blanco

Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
Gráficas Reunidas
Avda. Aragón, 56 (MADRID)
Fotocomposición
Novocomp, S.A.
Nicolás Morales, 38-40
Fotomecánica
GROF
Ezequiel Solana, 16
Depósito Legal:
M-5836-1986

Derechos exclusivos
de la revista
COMPUTING with the AMSTRAD

Representante para Argentina, Chile, Uruguay y Paraguay, Cia. Americana de Ediciones, S.R.L. Sud América 1.532. Tel.: 21 24 64. 1209 BUENOS AIRES (Argentina).

Se solicitará control OJD

MICROHOBBY AMSTRAD sumario

Año I • Número 2 • Junio 1986
Precio 475 ptas. Canarias, Ceuta y Melilla 450 ptas.

Impresoras



Si el mercado de ordenadores es una selva oscura para el más enterado, el de las impresoras es ya el caos más absoluto. Ponemos orden en este maremagnum y te damos la posibilidad de que elijas, con conocimiento de causa, la impresora que necesitas.



Para... PCW

Una de las posibilidades más potentes del Mallard Basic, sin duda, es la indexación de ficheros. Jetsam, constituye hoy por hoy, una de las formas más lógicas, eficaces y rápidas de indexar un fichero, apréndelas con nosotros y utiliza toda la versatilidad de tu PCW.

28 Serie oro

Hay gente, mucha, que no acepta la derrota de su inteligencia por un ordenador. Si quieres pasar a ser uno de ellos, ¡juega e intenta vencer a nuestro geniecillo.



Inteligencia artificial 40

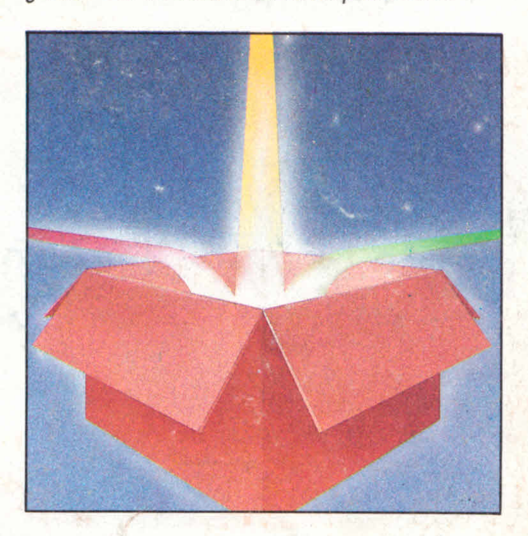
*Convertir un ordenador en algo inteligente, era hasta ahora muy difícil; si además este ordenador era uno modesto, la empresa era casi imposible. **AMSTRAD Especial,** continuando con el curso del semanario, os ofrece gratis el lenguaje A por excelencia: el LISP.*

*Utilizándolo, cualquiera que tenga un **Amstrad** y curiosidad por el asunto va a poder hacer «razonar» a su ordenador.*



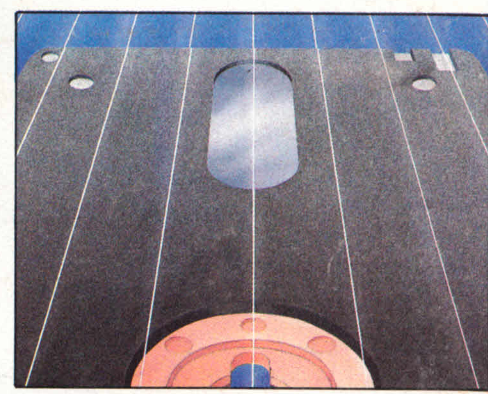
Soft integrado

Técnicas de programación verdaderamente depuradas y estudiadas han conseguido el milagro, tres programas en uno. Fidicom, un paquete de gestión con verdadera vocación profesional.



78 Paginación de memoria

*Elige el número de programas Basic que quieres y disponte a ejecutarlos, casi, casi, al mismo tiempo. Nuevos comandos RSX que consiguen convertir a tu **Amstrad** en dos, o en tres, o en...*



Pasatiempos 88

En broma y en serio, en castellano y en inglés, retamos a todo el poderío de tu ingenio para resolver nuestros xigramas. ¿Te atreves...?

COMPILADOR CBASIC

Daniel Palomo y Martín Carreira



Al poco de salir al mercado el sistema operativo CP/M, Gordon Eubanks Jr. reveló parte de su tesis doctoral en la que se describía un nuevo lenguaje: El EBasic.

EBasic fue el primer compilador para **Basic**. En realidad, el que hoy se nos ofrece, posee una estructura muy parecida al que se puede considerar fue su precursor.



l poco tiempo, Mr. Gordon Eubanks fundó su empresa y prosiguió su labor, ampliando el lenguaje por él ideado a lo que sería llamado **CBasic**. Había nacido un **Basic** insospechado...

Por aquel entonces, otra compañía americana, desarrolló otro dialecto Basic: El MBasic o Basic Microsoft. Hasta 1980 fue un lenguaje interpretado, fecha en la que apareció el primer compilador.

Desde que fue ideado en los años 60, el Basic fue enriqueciéndose, adquiriendo nuevos conceptos provenientes de otros lenguajes de alto nivel, y subsanando deficiencias. CBasic no se quedó atrás y el resultado es un lenguaje muy versátil y potente. El que sea compilado dice mucho en favor de su rapidez y puede ser utilizado por aquellos a los que el Basic se les ha quedado pequeño.

Más allá de la incorporación de nuevos comandos y funciones, que serán comentados más tarde, se entra en una nueva filosofía de programación. CBasic es para programar en módulos con estructuras, tal como hacemos en ciertos lenguajes de alto nivel como C, PASCAL, etc.

Esto lo conseguimos en CBasic con las funciones.

Funciones... y eso, ¿qué es?

Ante todo una función en **CBasic**, no es lo que estamos acostumbrados a ver, es algo mucho más potente; las funciones en CBasic, son como las de C, o procedures de Pascal.

Al igual que Basic, disponemos de una serie de funciones definidas (sin, sqr, etc.), y podemos crear nuevas funciones. La diferencia estriba en que nuestras funciones pueden contener fragmentos de programa. Algo parecido ocurre en la declaración de procedimientos de otros lenguajes. Esto permite tratar algoritmos complejos basados en técnicas de recursividad e iteración.

Podemos usar cualquier palabra reservada de **CBasic** en nuestras funciones, pero no las podemos tratar como si fuesen vulgares subrutinas, muy importante: a las funciones no las podemos invocar con GOTO o GOSUB, debiéndose utilizar el primero únicamente en casos desesperados. La palabra reservada que se encarga de invocar a una función .s CALL, pudiéndose invocar en expresiones con sólo su nombre. Para utilizar GOSUB debemos poner una etiqueta y el correspondiente RETURN en el cuerpo de la función, pero veremos que no nos será necesario utilizar subrutinas.

Un nombre de función es un identificador válido de **CBasic**. Solamente los seis primeros caracteres son significativos, como ocurre con el resto de los nombres de variables y etiquetas.

El tipo de función puede ser determinado con los identificadores adecuados, éstos son:

$ para cadenas de caracteres
% para enteros

el carácter final de una función o variable real no puede concluir con ninguno de los anteriores caracteres.

Podemos definir funciones como en Basic y tratarlas de igual manera.

Las funciones se pueden declarar como públicas o externas; las públicas son ejecutables por otros módulos de programa, únicamente nuestra declaración deberá tener como última palabra PUBLIC, para referenciar a estas funciones en un módulo de programa distinto, debemos utilizar otra función como puente, ésta estará declarada como EXTERNAL, y contendrá únicamente datos sobre la otra función, no puede contener comandos ejecutables.

Palabras reservadas

La lista completa de palabras reservadas de CBasic la tenemos en la figura 1. Haremos una descripción de aquéllas que supongan una innovación o, aun utilizando el mismo nombre, signifiquen acciones distintas que en el Basic Locomotive. Los gráficos en **CBasic** son muy potentes, por lo que serán tratados posteriormente.

Dos palabras reservadas por el sistema son ATTACH y DETACH y se refieren al control de la impresora por parte de sistemas multiusuario. ERRX es una función destinada al tratamiento de errores en MP/M II (multiusuario).

Veamos esas palabras nuevas:

DEF: Se utiliza para definir funciones en los sentidos descritos anteriormente. FEND identificará el final de dicha declaración. CALL llamará a la función siempre que no se la invoque en expresiones, en cuyo caso bastará con mencionar el identificador de la misma.

CHAIN: Sustituye un programa por otro y comienza su ejecución. Sirve para crear programas transitorios. Todas las variables que se quieren traspasar al programa llamado, deberán ser declaradas en una cláusula COMMON.

COMAND$: Se utiliza para transmitir parámetros a un programa.

CONCHAR%: Esta función espera un carácter del teclado y retorna su valor ASCII decimal, imprime el carácter.

INKEY: Espera un carácter del teclado y retorna su valor ASCII decimal, no imprime el carácter.

CONSOLE/LPRINTER: Dirigen la salida por pantalla o impresora respectivamente.

CONSTAT%: Retorna un valor booleano dependiendo del estado del teclado.

CREATE: Crea un fichero en disco, sin información en él. Permite determinar la longitud de registros, el número de buffers, el modo de fichero, siendo éstos:

> LOCKED: Ficheros a los que sólo puede acceder el programa que los crea.
> UNLOCKED: Ficheros utilizables por cualquier programa.
> READONLY: Ficheros de sólo lectura.

DELETE: Borra un fichero del disco dándole el número del mismo.

FLOAT: Convierte un número en un real de punto flotante.

GET: Lee un byte de datos de un fichero especificado.

IF END#: Transfiere el control del programa a una etiqueta determinada, cuando ocurre algún suceso de excepción en el acceso a un disco.

INITIALIZE: Permite cambiar los diskettes en tiempo durante la ejecución del programa, sin tener que inicializar el sistema.

LOCK: Previene al programa para modificar los datos en un registro.

MATCH: Busca subcadenas en cadenas retornando su posición.

OPEN: Abre un fichero ya creado. Permite determinar la longitud de los registros, el número de buffers, y el modo.

POS: Retorna la próxima columna en la que se va a imprimir.

PUT: Escribe un byte de datos en un fichero especificado.

RENAME: Cambia en el directorio del disco un nombre de fichero.

SADD: Retorna la dirección de comienzo de la variable de cadena especificada.

SHIFT: Retorna un entero que es aritméticamente desplazado un número de posiciones a la derecha.

SIZE: Retorna el número de bloques de un kilobyte, ocupados por el fichero especificado.

UCASE$: Convierte minúsculas a mayúsculas.

UNLOCK: Permite acceder y modificar los registros de un fichero.

VARPTR: Retorna la dirección de una variable.

Aquí finalizan las palabras reservadas, que nos pueden ser desconocidas. El resto no se comentan por no diferenciarse en nada de sus homólogas en **LOCOMOTIVE Basic**.

Conviene hacer algunas aclaraciones, sobre variables. Estas pueden ser declaradas en un tipo determinado, mediante los comandos:

INTEGER: Variables enteras.

REAL: Variables reales.

STRING: Variables de cadena.

COMMON: Variables de cualquier tipo que vayan a ser usadas por cualquier programa, llamado por **CHAIN**.

La declaración de variables nos evita tener que poner los identificadores de tipo.

En la declaración de variables debemos dejar un espacio en blanco entre el separador y la variable que lo precede.

Las etiquetas son tratadas como cadenas de caracteres.

Todos los errores en tiempo de ejecución, no tratados por el programa, son fatales y hacen que ésta concluya.

Las sentencias: DIM, DATA, IF, END y las declaraciones de variables, no podrán compartir la línea con ninguna otra palabra reservada.

El valor booleano verdadero es igual a —1.

FOR convierte todas las expresiones numéricas, que se hallan en el bucle, al tipo de variable que utiliza (real o entera).

Los símbolos para los formatos de PRINT USING son los mismos que los existentes en **LOCOMOTIVE Basic**.

JERARQUIA DE OPERADORES

1	()	Paréntesis
2	.	Potenciación
3	# /	Multiplicación y división
4	+—	Suma y resta
5	< , + , < =, + =, < + , =	Operadores de relación
6		NOT
7		AND
8		OR
9		XOR

Respecto a las funciones conviene resaltar ciertos aspectos: No pueden estar nidadas. Cualquier función puede llamar a otra. El comando COMMON no puede aparecer en una definición.

No puede haber GOTO's que referencien líneas fuera de la función.

Si incluimos sentencias DIM en una función, cada vez que ésta se ejecute, colocará una nueva matriz, que sustituirá a la anterior perdiendo los datos almacenados en ella.

Partes y funcionamiento

El compilador **CB80** consta de tres partes principales:

1. El compilador en sí,
2. El linker y
3. La librería.

Vamos a hacer aquí una descripción de las partes, así como alguna particularidad del funcionamiento.

El compilador

Es el encargado de traducir el programa «fuente» escrito en **CBasic** a un programa en máquina relocalizable. Algunas instrucciones son llevadas a un código intermedio para ser traducidas totalmente durante el proceso de linkado.

Para que esto sea posible, el compilador crea tres ficheros intermedios llamados PA.TMP, CODE.TMP y DATA.TMP. Los tres son borrados automáticamente por el compilador cuando termina su cometido.

Por último, se grabará en el disco un pseudoprograma en código máquina con la extensión .REL. Este archivo es relocalizable y es el que precisamos para su posterior linkado.

En esta fase se pueden presentar tres tipos distintos de errores, que deberán ser corregidos antes de proseguir:

1. Error en los espacios de memoria o de ficheros: Ocurre cuando sobrepasamos la memoria disponible, el disco está lleno, comandos de línea incorrectos, error de lectura del disco, intentamos leer un fichero no abierto previamente,... etc.

2. Errores de Compilación: Corresponden a mal uso de un comando, caracteres no válidos, declaraciones incorrectas u omisión de delimitadores. Por cada error detectado se escribe una (↑) indicando el lugar del texto fuente, y un código variable entre 1 y 117 que informa de la naturaleza del mismo. Cabe decir, que un solo error puede generar varios códigos, incluso en otras zonas del programa.

3. Errores fatales: Teóricamente no deben producirse y a nosotros no nos ha sucedido. *Digital Research* está interesada en el programa que motiva un error de este tipo, así como del código de error que genera.

Opciones de comando directo

Deben ser insertadas en una línea del programa fuente sin que haya en ella comandos o funciones **CBasic**. No pueden ser etiquetadas y no dan lugar a ningún código ejecutable. Es una manera de seleccionar, mediante programa, opciones del compilador. Son las siguientes:

%LIST, %NOLIST: Controlan el que se produzca o no el listado del código fuente durante la fase de compilación.

%EJECT: Hace que el listado por impresora continúe al principio de una nueva página.

%PAGE: En la impresora, selecciona la longitud de página.

%INCLUDE: Incluye el código fuente almacenado en otro archivo durante la compilación de un programa.

%DEBUG: Controla mediante tres indicadores la aparición o no del código objeto mezclado con el fuente (I), genera o no el código del número de línea (N) y coloca o no el número de línea en el fichero del código fuente (V).

Se dispone, además, de otros 12 conmutadores (banderas), que permiten seleccionar otras tantas modificaciones en el control de periféricos y sistema operativo.

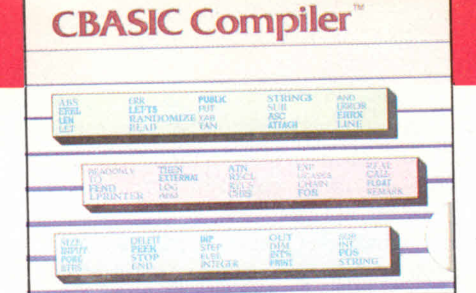
Los programas que utilicen gráficos deberán incluir los siguientes ficheros:

GRAFCOM.BAS
CIRCOM.BAS

El último fichero no es necesario si no se incluyen funciones de círculo.

El linker

La misión del compilador era traducir un programa fuente en un fichero en código máquina relocalizable. Algunas instrucciones quedaban en un código intermedio que será to-



talmente traducido mediante llamadas a la librería.

La función del linker o **«enlazador»** es precisamente la de encadenar entre sí los distintos módulos de programa. Asimismo se encargará de llamar a la librería para añadir los módulos designados por el **«código intermedio»** que no traducía completamente el compilador. Por último, creará las rutinas necesarias para el control y ejecución del programa. Al final dispondremos de un fichero ejecutable (bajo cp/m) con la extensión .COM.

Al igual que el compilador, informa en hexadecimal del tamaño, comienzo y final del programa objeto, el área de datos y del área de variables comunes.

Opciones del linker

Mediante cinco conmutadores podemos efectuar alteraciones en los ficheros grabados o dirigir la salida por impresora.

La más interesante de todas es la [S] que, una vez activada permite utilizar la técnica de recubrimiento (overlay). Consiste en una zona fija de memoria donde están almacenadas distintas rutinas en momentos distintos. El aprovechar esta técnica supone aceptar restricciones tales como que el tamaño de todas las rutinas sea el mismo, que las distintas rutinas no llamen a funciones de librería distintas,... etc. Su uso requiere cuidado y puede ser útil cuando la longitud del programa exija «rentabilizar» memoria.

La librería

Consta de una serie de rutinas contenidas en un fichero indexado. El linker mezclaba estas rutinas con la secuencia generada por el compilador. Cabe destacar la posibilidad de crear una librería indexada definida por el usuario y concatenar ficheros de este tipo.

Gráficos

Los gráficos en **CBasic** son una herramienta muy potente, nos permiten hacer cualquier cosa que se nos ocurra.

Las funciones y comandos de gráficos se dividen en cinco grupos dependiendo de la función que realicen. Los comandos y funciones de gráficos se consideran palabras reservadas de CBasic, solamente si nuestros programas usan gráficos.

La lista de palabras clave es la siguiente:

GRUPO	PALABRAS CLAVE
SALIDA	GRAPHIC PRINT MAT FILL MAT MARKER MAT PLOT PLOT
FORMATEO	CHARACTER HEIGHT (SET/ASK) COLOR (SET/ASK) COLOR COUNT (SET/ASK) JUSTIFY (SET/ASK) LINE STYLE (SET/ASK) MARKER HEIGHT (SET/ASK) MARKER TYPE (SET) STYLE COUNT (ASK) TEXT ANGLE (SET/ASK)
AREA DE PANTALLA	BOUNDS (SET/ASK) DEVICE (ASK) VIEWPORT (SET/ASK) WINDOW (SET/ASK)
ENTRADA	GRAPHIC INPUT
CONTROL	BEAM (SET/ASK) CLEAR CLIP (SET/ASK) GRAPHIC CLOSE GRAPHIC OPEN POSITION (SET/ASK)

Para trabajar con gráficos en **CBasic** hay que tener muy claros los siguientes conceptos:

COORDENADAS: El dispositivo de la pantalla de gráficos, está definido por los ejes de coordenadas X e Y. Ambas comienzan en la esquina inferior izquierda y originalmente su rango va de 0.0 a 1.0. Esta disposición puede parecer extraña, pero podemos variar las escalas a nuestro gusto como veremos más adelante.

coordenadas iniciales

0.0,1.0	1.1,1.1
0.0,0.0	1.0,0.0

LIMITES (bounds): Son las dimensiones físicas en pixels, pulgadas, centímetros, o cualquier otro tipo de medida. Normalmente los dos ejes no suelen ser iguales.

Inicialmente la extensión de la pantalla es del 100% de la misma pudiendo ser reducido a nuestro gusto.

VIEWPORT: Es el área de pantalla en la cual se imprimen los gráficos. Este comando nos permite reducciones de dicho área a nuestro gusto.

WINDOW: Esta palabra reservada no tiene nada que ver con la de LOCOMOTIVE Basic. En **CBasic** sitúa la escala de los ejes de coordenadas con los que se desea trabajar.

BEAM: Activa o desactiva el modo dibujar. Equivale a PEN ON/OFF del plotter.

SET BEAM: Determina el estado (ON/OFF).

ASK BEAM: Devuelve en una variable de cadena el estado de BEAM.

BOUND: Determina las dimensiones físicas de la pantalla.

SET BOUND < alto + < ancho + : Determina las dimensiones de la pantalla.

CHARACTER HEIGHT: Define la altura de los caracteres relativa a la longitud del eje Y.

SET CHARACTER HEIGHT: Determina la altura dependiendo del número que vaya a continuación.

ASK CHARACTER HEIGHT: Retorna en una variable el valor vigente en ese momento.

CLEAR: Equivale a CLS.

CLIP: Activa o desactiva el control de los márgenes, nos permite dibujar figuras que excedan los límites de la pantalla.

SET CLIP: ON activa, OFF desactiva.

ASK CLIP: Retorna en una variable el estado de clip en la forma antes expresada.

COLOR: Asigna un color para texto o gráficos.

SET COLOR: Asigna un color determinado por un entero.

ASK COLOR: Retorna en una variable numérica el color actual.

DEVICE: Devuelve los límites de la pantalla en dos variables.

ASK DEVICE < var alto + < var ancho +

GRAPHIC INPUT: Permite posicionar el cursor con las teclas correspondientes y cuando se pulsa una que no sea de cursor, devuelve la posición del cursor en dos variables, en el orden X e Y, y el carácter generado por la tecla pulsada, es devuelto en una variable de cadena.

GRAPHIC OPEN: Inicializa el sistema de gráficos con posibilidad de determinar el dispositivo de salida.

(Consultar la guía de GSX para mayor información.)

GRAPHIC PRINT AT (X,Y): Sitúa en las coordenadas indicadas una constante o variable de cadena.

LINE STYLE: Selecciona el tipo de línea para dibujar.

SET LINE STYLE: Selecciona el tipo de línea dependiendo de los siguientes valores:

1. continua
2. rayada
3. punteada
4. punto-raya.

ASK LINE STYLE: Devuelve en una variable entera el tipo de línea existente en ese momento.

MARKER HEIGHT: Determina la altura de los marcadores con respecto al eje Y.

SET MARKER HEIGHT: Determina la altura dependiendo de una expresión numérica.

ASK MARKER HEIGHT: Devuelve en una variable entera el valor actual.

SET MARKER TIPE: Selecciona el tipo de marcador a utilizar dependiendo de la siguiente lista:

1 .
2 +
3 *
4 0
5 ×

MAT FILL: Dibuja un polígono relleno dependiendo de los valores actuales de las matrices de coordenadas X e Y.

MAT MARKER: Marca los puntos señalados por los valores actuales de las matrices de coordenadas, utilizando el tipo de marcador en curso.

MAT PLOT: Une los puntos referenciados por la matriz de coordenadas.

PLOT: Conecta con líneas una serie de pares de coordenadas dado.

POSITION: Selecciona la posición del cursor.

SET POSITION: Selecciona la posición del cursor por medio de dos enteros dados.

ASK POSITION: Devuelve la posición del cursor en dos variables.

ASK STYLE COUNT: Asigna a una variable el número del tipo de línea actual.

TEXT ANGLE: Determina el ángulo, con respecto al eje de las X, en el que se va a insertar texto.

SET TEXT ANGLE: Selecciona el ángulo, expresado en radianes.

ASK TEXT ANGLE: Retorna el valor del ángulo en una variable real.

VIEWPORT: Dispone los valores del área de pantalla en el cual se van a imprimir gráficos.

SET VIEWPORT: Selecciona las dimensiones de la pantalla, dándole, en este orden, la posición izquierda, derecha, abajo y arriba.

ASK VIEWPORT: Devuelve en cuatro variables las dimensiones actuales de la pantalla.

WINDOW: Determina la escala de los ejes X e Y.

SET WINDOW: Determina la escala dándole, en este orden, límite izquierdo, límite derecho, límite inferior, límite superior.

ASK WINDOW: Devuelve en variables, en el orden indicado, la escala actual.

Los gráficos en CBasic son rápidos y bastantes buenos, el disco contiene dos programas de prueba DEMOGRAF.BAS y TSTCIR,BAS, éstos los podemos compilar normalmente.

Pero antes de ejecutar el programa .COM, deberemos ejecutar GENGRAF, suministrado con CP/M plus, para mezclar con el programa un cargador en tiempo de ejecución de GSX,SYS. El sistema es el siguiente: GENGRAF PROG.

GENGRAF necesita un fichero .COM para operar.

Conclusiones

Un lenguaje como el aquí descrito, con capacidad de programación modular y estructurada es el ideal para pasar de los conocimientos actuales de Basic a una nueva dimensión del mismo, que nos permitirá el acceso a otros lenguajes de alto nivel. Por si fuera poco, su capacidad de manejo numérico supera incluso el de otros compiladores cuya especialización está más comprometida.

Tal es el caso de algunos disponibles para **Amstrad** como el Pascal, Fortran y el propio Locomotive. A efectos de dar cifras, ahí van:

Desde 10 ^—64 hasta 9.9999999999999 # 10 ^62. Realmente notable si tenemos en cuenta que la mantisa tiene 14 dígitos significativos.

CBasic es un lenguaje potente y rápido.

Para demostrarlo hemos hecho algunas pruebas:

BUCLE VACIO: desde 1 a 100.000 no mensurable. Desde luego menos de 1 segundo...

CALCULO E IMPRESION DE LOS 50 PRIMEROS NUMEROS NATURALES: 25 sg.

BUCLE CON REALIZACION DE 300 SUMAS E IMPRESION: 18 sg.

TIEMPO DE COMPILACION (CB80): 38 sg.

TIEMPO DE ENLAZADO (LK80): 31 sg.

CLASIFICACION DE BURBUJA 50 NUMEROS: en imprimir matriz sin clasificar, ordenar e imprimir matriz clasificada: 4 sg.

TIEMPO DE COMPILACION (CB80): 23 sg.

TIEMPO DE ENLAZADO (LoK880): 24 sg.

Los listados de los programas se incluyen a continuación:

```
REM CLASIFICACION POR BURBUJA
REM POR DANIEL PALOMO Y MARTIN
CARREIRA
DIM A(10)
PRINT CHR$(12)
RANDOMIZE
FOR B%=1 TO 10
A(B%)=INT (RND#100)
NEXT B%
DEF IMPRIME
DEF IMPRIME
DEF IMPRIME
DEF IMPRIME
FEND
DEF BURBUJA
        PRINT "CLASIFICANDO"
        FOR D%=1 TO 9
        FOR C%=1 TO 9
        IF  A(C%) + A(C%+1)
        THEN T=A(C%) :E
        A(C%)=A(C%+1):E
        A(C%+1)=T
        NEXT C%
        NEXT D%
FEND
CALL IMPRIME
CALL BURBUJA
PRINT "LISTA ORDENADA"
CALL IMPRIME
END
```

EQUIPO MINIMO

128 K de memoria RAM.
CPM 2.2 .PLUS sin utilizar gráficos.
CPM PLUS para gráficos.

CBASIC: PALABRAS RESERVADAS

ABS	ERR	LE	PUBLIC	STRING$
AND	ERRL	LEFT$	PUT	SUB
AS	ERROR	LEN	RANDOMIZE	TAB
ASC	ERRX	LEFT	READ	TAN
ATTACH	EQ	LINE	READONLY	THEN
ATN	EXP	LOCK	REAL	TO
BUFF	EXTERNAL	LOCKED	RECL	UCASE$
CALL	FEND	LOG	RECS	UNLOCK
CHAIN	FLOAT	LPRINTER	REM	UNLOCKED
CHR$	FOR	LT	REMARK	USING
CLOSE	FRE	MATCH	RENAME	VAL
COMMANDS	GE	MFRE	RESTORE	VARPTR
COMMON	GET	MID$	RETURN	WEND
CONCHAR%	GO	MOD	RIGHT$	WHILE
CONSOLE	GOSUB	NE	RND	WIDTH
CONSTAT%	GOTO	NEXT	SADD	XOR
COS	GT	NOT	SGN	%CHAIN
CREATE	IF	ON	SHIFT	%DEBUG
DATA	INITIALIZE	OPEN	SIN	%EJECT
DEF	INKEY	OR	SIZE	%INCLUDE
DELETE	INP	OUT	SQR	%LIST
DETACH	INPUT	PEEK	STEP	%NOLIST
DIM	INT	POKE	STOP	%PAGE
ELSE	INT%	POS	STR$	
END	INTEGER	PRINT	STRING	

```
REM BUCLES
%DEBUG VI
DEF BUCLE.VACIO
INTEGER I, N
PRINT "BUCLE VACIO"
I=INKEY
FOR N=1 TO 100000
NEXT N
I=INKEY
FEND

DEF BU.COS
PRINT "BUCLE COSENO"
I=INKEY
FOR N=1 TO 100
PRINT COS(N); " ";
NEXT N
I=INKEY
FEND

DEF SUMA
INTEGER I, N, AL, M, SUM, SUM1
PRINT "SUMA "
I=INKEY
RANDOMIZE
FOR N=1 TO 99
FOR M=1 TO 6
AL=INT (RND#1000)
SUM=SUM+AL
NEXT M
SUM1=SUM1+SUM
PRINT SUM,SUM1
NEXT N
I=INKEY
FEND

PRINT CHR$(12)
CALL BUCLE.VACIO
PRINT CHR$(12)
CALL BU.COS
PRINT CHR$(12)
CALL SUMA
END
```

EL MEJOR AMIGO DEL AMSTRAD

Antonio J. de la Cuadra

Para cualquier uso «en serio» que se le pretenda dar al ordenador será imprescindible contar con una impresora. La impresora es, pues, un periférico que nos servirá de gran ayuda para listar programas, obtener gráficos o incluso para convertir al ordenador en una práctica máquina de escribir con la ayuda de un «procesador de textos».

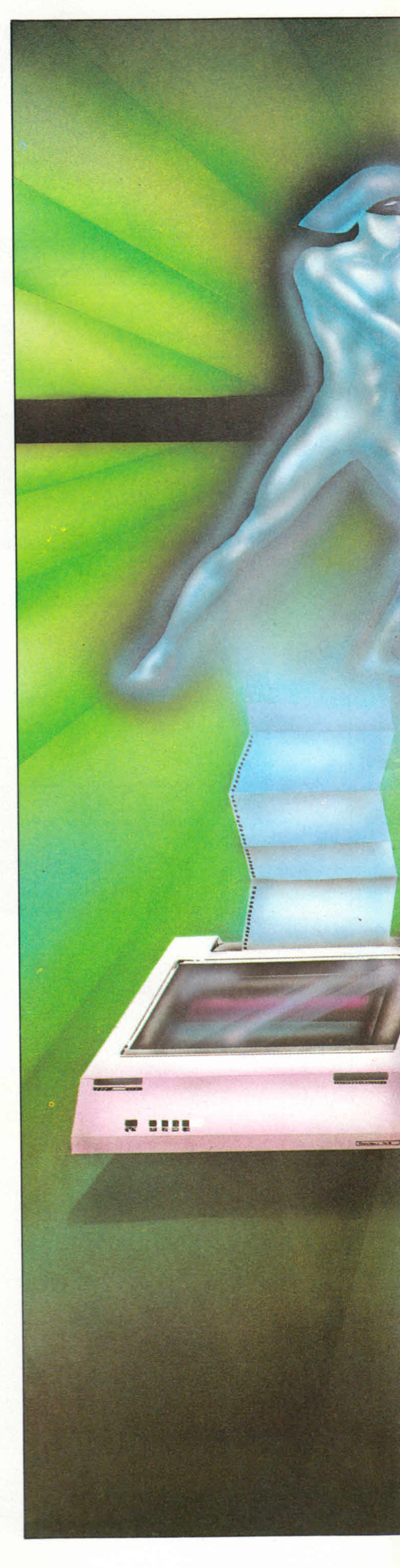
Existen infinidad de tipos de impresora según la forma que tienen de escribir los caracteres: matricial, de impacto o margarita, trazadores o plotters, y láser. Las primeras cuentan con una cabeza compuesta por un conjunto de puntos que componen el carácter solicitado, de forma similar a como aparecen en la pantalla de vuestro ordenador. Las impresoras de margarita (**«daisy»**) ya tienen definido el carácter como en una máquina de escribir convencional y obviamente con ellas se obtiene una mayor calidad de impresión pero, por contrapartida, no pueden jugar con las diferentes posibilidades de tamaños de los caracteres, ni obtener «soft copys» de pantalla ni gráficos definidos por el usuario. Los plotters están más orientados a una utilización de trazadores de curvas y aunque permiten «dibujar» caracteres, su aplicación para procesadores de textos no es nada recomendable puesto que «escriben» a una velocidad excesivamente lenta *(del orden de 5 caracteres por segundo, 5 cps)*. Por último, las impresoras láser representan el último avance de la tecnología en este sector y, a pesar de sus grandes ventajas de rapidez y alta calidad de caracteres y gráficos por la impresión de un rayo láser sobre el papel, resultan prohibitivas para el usuario medio por su encarecido precio.

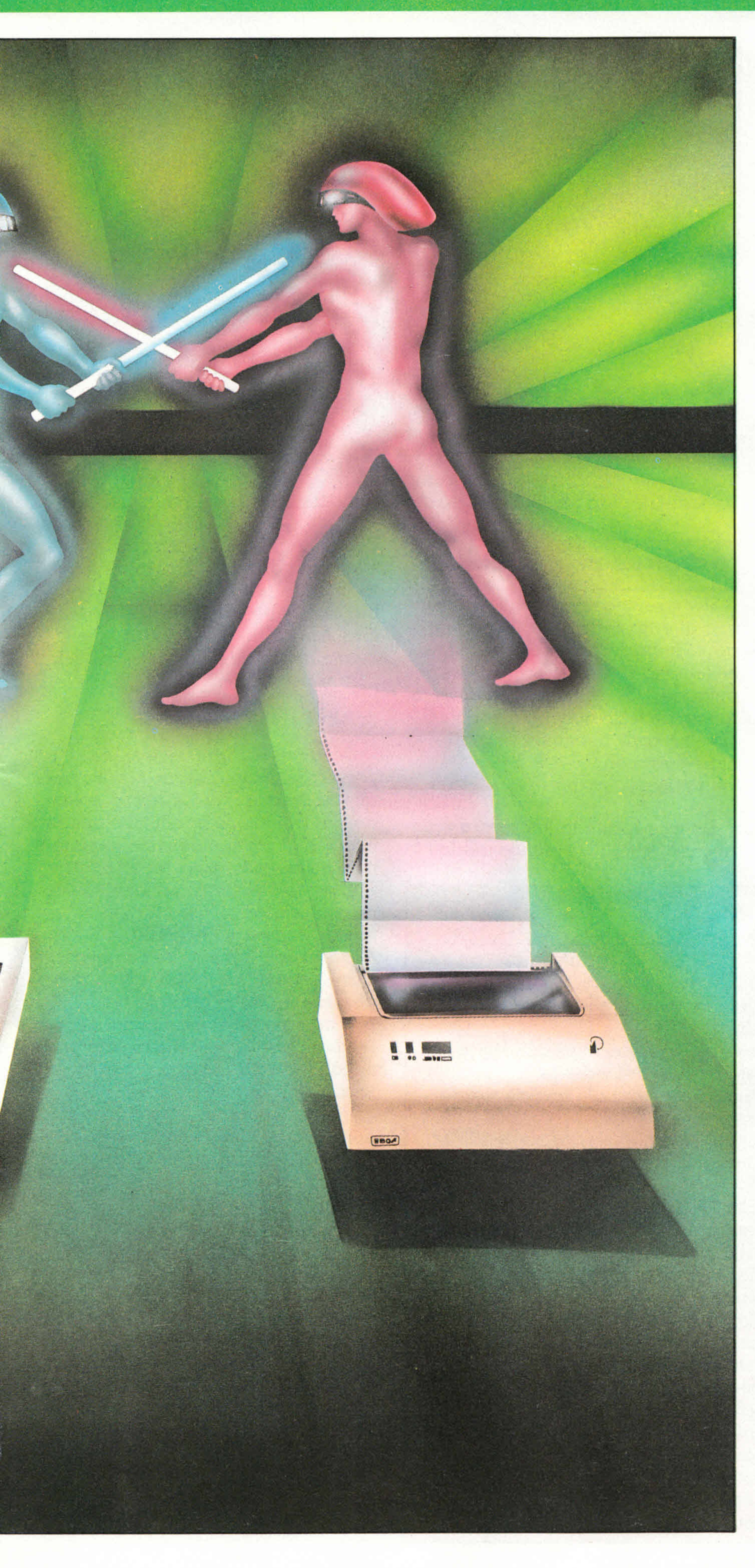
Por sus posibilidades y su buena relación precio/calidad nos vamos a centrar en el presente dossier en las impresoras del tipo matricial *(Dot Matrix)*. Aunque podamos encontrarlas en el mercado desde las 20.000 ptas., el precio de una impresora de este tipo de calidad media suele oscilar por las 50.000 pesetas, y con ella no sólo nos permitirá redactar informes y listar programas sino que además dispondremos de una pequeña imprenta capaz de jugar con distintos tipos de letra y combinarlas con caracteres gráficos.

Conector «Centronic» casi completo

El **Amstrad** CPC se comunica con la impresora a través de un conector «Centronic». Este tipo de conector se diferencia del conocido por RS-232 en que los datos que envía el ordenador a su periférico circulan en «paralelo» o simultáneamente a través de cables independientes. Por el contrario, en el conector RS-232 los bits recorren el mismo cable uno detrás de otro o en **«serie»;** por esta razón este conector es el que utiliza para comunicaciones vía MODEN entre ordenadores.

La impresora funciona con unos códigos de caracteres que varían entre 0 y 255, que traducidos en siste-





ma binario son respectivamente &X00000000 y &X11111111, por lo que para definirlos necesitaremos un total de 8 bits, o sea, 8 cables en el ordenador Centronic. El resto de los contactos del Centronic lo forma el sistema de «protocolo», esto es las normas que deben cumplirse para comunicarse el ordenador con la impresora.

Sin embargo, por una extraña razón, el conector Centronic del **Amstrad** CPC no está completo, y el bit más significativo se encuentra puesto siempre en estado bajo (0) variando los códigos entre &X00000000 y &X01111111, por lo que desde nuestro ordenador favorito sólo dispondremos de los 127 primeros caracteres de la impresora, afortunadamente los que componen el código ASCII (Código Americano Standard para el Intercambio de Información), y perderemos en principio el segundo juego de caracteres dedicado generalmente a gráficos y alfabeto griego.

Y decimos en principio porque no está todo perdido. Para los afortunados poseedores de una impresora Seikosha, Admate o New Print, les será posible acceder al escondido segundo juego de caracteres mediante:

```
PRINT#8, CHR$(27); ''='';
```

para la Seikosha SP-1000 CPC, o bien:

```
PRINT#8, CHR$(27); ''?O'';
```

para los modelos Admate/New Print DP-100/DP-140/DP-80/CPA-80. De esta forma levantamos el bit más significativo al estado alto (1) y accederemos a los códigos altos. Para reponer el estado normal basta con hacer:

```
PRINT#8, CHR$(0);
PRINT#8, CHR$(27); ''?I'';
```

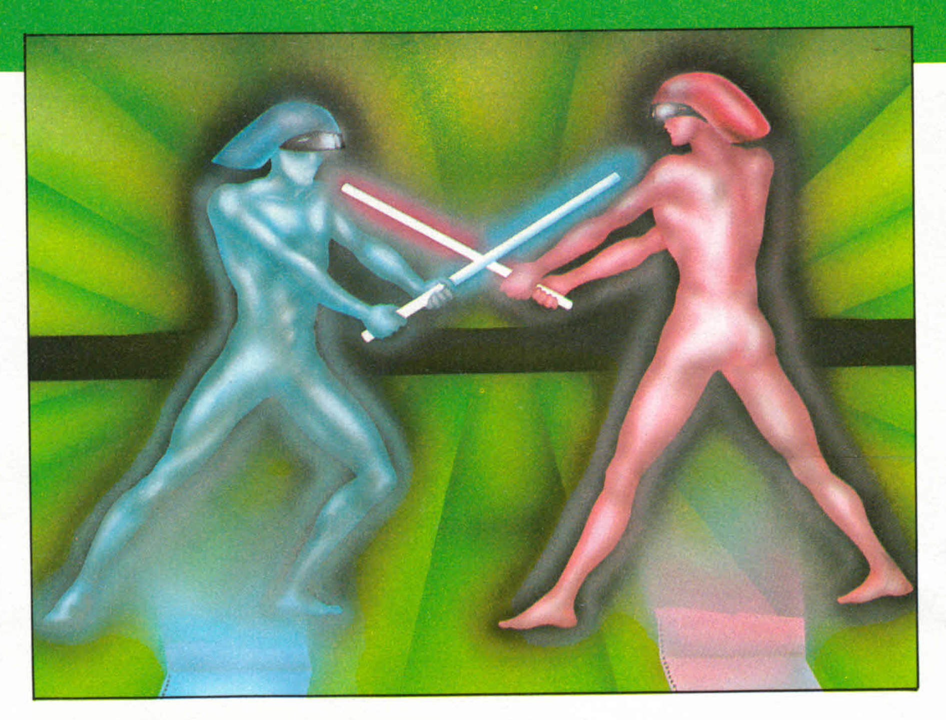
Para la Seikosha y Admate/New Print respectivamente.

El resto de usuarios de impresoras lo tendrá más difícil, aunque no imposible, puesto que en Inglaterra la firma Hitech ha creado un periférico que permite acceder al segundo juego de caracteres incluyendo además «copys» de pantalla mediante comandos RSX. El kit hardware y software se encuentra en el mercado británico al precio de 15.75 libras y es de esperar que en un futuro algún distribuidor nacional se haga cargo de su importación.

Impresoras con RAM

Ya hemos entrado en el comando para la impresora: ''PRINT#8,'', o bien en forma abreviada ''?Pt8,''. De ésta podemos ordenar a la impresora que escriba una frase, por ejemplo, ''PRINT#8, **MICROHOBBY AMSTRAD**''. Comprobarás que si no tienes encendida la impresora, el ordenador se quedará colgado sin devolverte el «Ready». Esto se debe a que el sistema de protocolo no devuelve el control al ordenador hasta que se haya cumplido la instrucción. Estas instrucciones pasan previamente por un «buffer» o tampón de memoria que no es más que una especie de cámara de descompresión que almacena los datos antes de ser impresos.

Habrás comprobado que cuando se escriben listados de programas largos con el comando ''LIST#8'', el ordenador se queda inmovilizado a la espera de terminar el listado. Opcionalmente en la mayoría de las impresoras que conocemos, el propietario puede adquirir una ampliación del buffer de RAM que evita la situación mencionada, posibilitando la utilización del ordenador mientras que trabaja la impresora.



Infinitas posibilidades

Siempre que se reinicialice y mientras que no se le ordene lo contrario, la impresora escribirá caracteres en tamaño «PICA» (10 caracteres por pulgada). Si deseamos cambiar el tamño de letra al denominado **«ELITE»** (12 cpp) debemos preparar la impresora con:

PRINT#8, CHR$ (27); ''M'';

De esta forma la impresora reconoce el CHR$ (27) (denominado EScape) y se dispone a cambiar uno de sus formatos, en este caso al detectar la letra **'M'** —CHR$ (77)— cambia a un tipo de letra ligeramente más estrecha. Para volver nuevamente al tipo de letra anterior basta con hacer:

PRINT#8, CHR$ (27); ''P'';

Debemos recordar que existen impresoras que no cuentan con las posibilidades que citamos en el presente dossier por lo que por mucho que se intenten los comandos que exponemos no se conseguirá nada.

Otro tipo de letra con el que puede contar nuestra impresora es el conocido como **'PROPORCIONAL'**. Aunque este tamaño de carácter es el mismo del tipo **'PICA'** no se puede decir en este caso que tenga 10 cpp puesto que el espaciado de caracteres es inversamente proporcional al ancho de la letra. Así pues, la separación en los caracteres 'w' o 'm' será menor que la de 'i' o '!'. Suponiendo que contaramos con este tipo de caracteres bastaría para activarlo o desactivarlo respectivamente con:

PRINT#8, CHR$ (27); ''p''; CHR$ (n);

con n=0 para desactivarlo y n=1 para activarlo. Hay que resaltar que el carácter 'p' debe estar en minícula.

Con estos tres tipos de letra tenemos la posibilidad de **'CONDENSAR'** los caracteres convirtiendo el paso **'PICA'** que un cuerpo de letra de 17 cpp. Para ello, debemos hacer lo que se conoce por 'SI':

PRINT#8, CHR$ (15);

CODIGOS DE CONTROL

Denom.	Decimal	Hexadec.	Función
NUL	CHR$ (0)	CHR$ (&0)	Fin de tabulador con ESC D
BEL	CHR$ (7)	CHR$ (&7)	Suena la chicharra de la impresora durante 0,3 seg.
BS	CHR$ (8)	CHR$ (&8)	Retrocede el cabezal un espacio. Muy útil para poner acentos y diéresis
HT	CHR$ (9)	CHR$ (&9)	Desplaza el cabezal a la siguiente posición marcada por el tabulador horizontal
LF	CHR$ (10)	CHR$ (&A)	Salta una línea
VT	CHR$ (11)	CHR$ (&B)	Tabulación vertical. Si no está fijada hace un LF
FF	CHR$ (12)	CHR$ (&C)	Salta una página
CR	CHR$ (13)	CHR$ (&D)	Retorno de carro. Imprime el contenido del buffer y lo vacía
SO	CHR$ (14)	CHR$ (&E)	Modo expandido hasta el final de línea o recepción de DC4
SI	CHR$ (15)	CHR$ (&F)	Modo comprimido. Permanece hasta que se cancela con DC2
DC1	CHR$ (17)	CHR$ (&11)	Pone la impresora en 'SELECT'
DC2	CHR$ (18)	CHR$ (&12)	Termina caracteres comprimidos y vacía el buffer
DC3	CHR$ (19)	CHR$ (&13)	Coloca la impresora en modo 'DESELECT'
DC4	CHR$ (20)	CHR$ (&14)	Termina el modo expandido
CAN	CHR$ (24)	CHR$ (&18)	Borra todos los caracteres del buffer
ESC	CHR$ (27)	CHR$ (&1B)	Código 'ESCAPE'. Prepara la impresora para recibir una secuencia de control

Para su desconexión tendremos que hacer el control 'DC2' que es:

PRINT#8, CHR$ (18);

Del mismo modo para obtener un carácter **'EXPANDIDO'** debemos hacer un control 'SO' y para desactivarlo el conocido por 'DC4'. Estos dos controles se programan respectivamente con:

PRINT#8, CHR$ (14); PRINT#8, CHR$ (20);

Otro punto que puede resultar de gran interés son los caracteres **'SUPERINDICES'** y **'SUBINDICES'.**

Estos tipos de impresión se logran comprimiendo a la mitad la altura de un carácter normal.

Para activarlos debemos hacer:

PRINT#8, CHR$ (27); ''S''; CHR$ (n);

donde n debe ser 1 para **'SUBINDICES'** y 0 para **'SUPERINDICES'.** Para desactivar cualquiera de los modos:

PRINT#8, CHR$ (27); ''T'';

El 'SUBRAYADO' de los caracteres y su posterior desactivación se consigue respectivamente con:

PRINT#8, CHR$ (27); ''—''; CHR$ (1);
PRINT#8, CHR$ (27); ''—''; CHR$ (0);

La letra en **'NEGRITA'** se consigue haciendo una segunda pasada a la misma altura con una separación de 1/120 de pulgada. Este modo no es compatible con los caracteres **'SUPERINDICES'** y **'SUBINDICES'.**

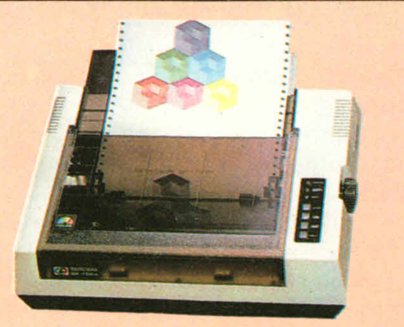
Para activarlo o desactivarlo debemos hacer:

PRINT#8, CHR$ (27); ''E'';
PRINT#8, CHR$ (27); ''F'';

El modo de 'REPICADO' es similar al de la 'NEGRITA' pero la segunda pasada se hace por encima de la primera a una altura de 1/144 de pulgada. Su activación o desactivación se consigue con:

PRINT#8, CHR$ (27); ''G'';
PRINT#8, CHR$ (27); ''H'';

Hasta aquí prácticamente está todo dicho sobre los diferentes tipos de letra. No obstante existen impresoras con un tipo de letra llamado **'NLQ'** o de **'ALTA CALIDAD';** con ello se permite un estilo de impresión con mucha mejor definición que el modo estándar.



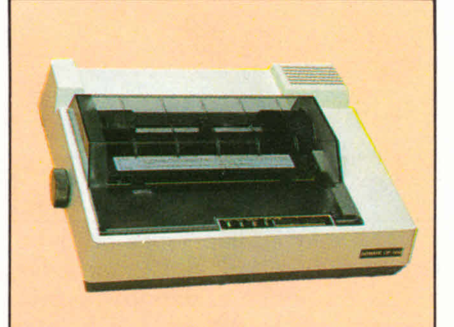
SEISKOSHA GP-700 A COLOR
Matriz de impresión de 7×8. Velocidad de impresión 50 cps. 80 columnas. Caracteres normales y expandidos, gráfica, cuatro tipo de caracteres. Tracción/Fricción. Impresión a color.
Precio .... 72.688 ptas.
Cartucho color .... 3.125 ptas.
Cartucho negro ...... 1.781 ptas.



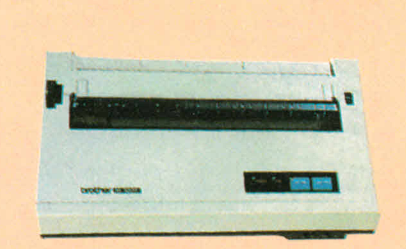
SEISKOSHA GP-50 A
Matriz de impresión de 5×8. Velocidad de impresión 40 cps. 46 columnas. Caracteres normales y expandidos, gráfica. Sólo fricción. Alimentador de corriente externo.
Precio .... 22.288 ptas.
Cartucho tinta .... 1.568 ptas. (*)
(*) En colores rojo, verde, negro, azul, marrón, morado, naranja.

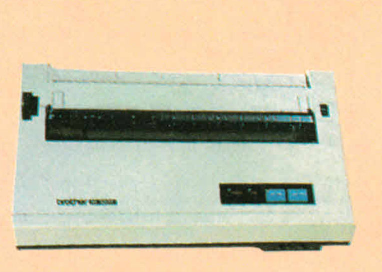


SEISKOSHA SP-1000 CPC
Matriz de puntos de alta velocidad (100 cps), modo de impresión bidireccional y optimizada, capacidad gráfica con alta resolución. 80 columnas y 137 en comprimido. NLQ a 24 cps. Tracción y fricción. Introductor automático hoja a hoja. Preparada para Centronic de 7 bits
Amstrad.
Precio .... 72.688 ptas.
Cartucho tinta .... 1.568 ptas.



NEW PRINT/ADMATE DP 100
Matriz de puntos de 8×8. Velocidad de impresión 100 cps. Impresión bidireccional. Tracción y fricción. 80 columnas en PICA y 142 en comprimido. Preparada para Centronics de 7 bits.
Amstrad
Precio .... 62.600 ptas. (*)
Cartucho tinta .... 1.120 ptas.
(*) Mismas características con 80 cps para DP 80 LQ (51.400) y CPA 80 con 80 cps y NLQ (62.600 ptas.).





BROTHER M-1009
Velocidad de impresión 50 cps. Impresión bidireccional, unidireccional para gráficas. Tracción y fricción. Introductor hoja a hoja.
Precio .... 44.000 ptas.
Cartucho tinta .... 516 ptas.
(*) Existe la versión M-1109 con 100 cps y las mismas características por 51.500 ptas.

C.ITOH 7500 AP
Velocidad de impresión 105 cps. Tracción y fricción. 80 columnas en pica y 136 en comprimido. Buffer de 2 K. Gráficos por bit imagen.
Precio .... 94.528 ptas.
Cartucho tinta .... —

Otra posibilidad que podemos conseguir con las impresoras es variar como **'LINE FEED'** (LF). Nuevamente nos encontramos con el sistema de medida inglesa en pulgadas que es el que parece que se toma como estándard. A continuación exponemos los comandos con su respectiva consecuencia:

PRINT#8, CHR$ (27); ''0'';	LF a 1/8 ''
PRINT#8, CHR$ (27); ''1'';	LF a 7/72 ''
PRINT#8, CHR$ (27); ''2'';	LF a 1/6 ''
PRINT#8, CHR$ (27); ''3''; CHR$ (n);	LF a n/216''
PRINT#8, CHR$ (27); ''A''; CHR$ (n);	LF a n/72 ''

A corazón abierto

Desmontar la impresora no debe causar ningún trauma al usuario. De hecho los manuales dan los oportunos consejos para hurgar en su interior donde precisamente se encuentra un conjunto de microinterruptores que permiten fijar una serie de condiciones como el salto de página o **'FORM FEED'** (FF), el salto de línea, la cantidad de columnas por línea, y el conjunto de caracteres. Esto nos puede venir que ni pintado si el papel continuo que hemos escogido no está sincronizado con el salto de página, o bien queremos fijar los caracteres en castellano.

Habréis podido comprobar las enormes posibilidades de las impresoras, sus precios están en función de la cantidad de formatos de impresión. Si todavía no os habéis decidido nuestro catálogo os servirá de gran ayuda. Si ya la tenéis es nuestro deseo que el presente dossier os valga de gran ayuda.

Consejos

- Conecta el cable de la impresora al ordenador con ambos apagados.
- Conecta primero el ordenador y posteriormente la impresora.
- Asegúrate que el voltaje que utilizas es el que especifica el fabricante.
- No toques el cabezal mientras que está imprimiendo ni después de imprimir.
- Cuando utilices papel continúa, asegúrate que los agujeros están alineados.
- Evita doblar la cinta cuando la instales.
- Una vez apagada la impresora no la enciendas hasta pasados dos segundos.
- No coloques el tractor cuando está en modo de fricción.
- No intentes imprimir sin papel ni cinta.
- Efectúa periódicamente una limpieza de polvo en el interior de la impresora.
- No hagas caso de estos consejos, si el manual especifica lo contrario.

PROGRAMA 1

```
10 PRINT#8,CHR$(27);CHR$(64);:REM R
esetea la impresora.
20 PRINT#8,CHR$(15);"Cuerpo 'CONDEN
SADO': 17 caracteres por pulgada.":
REM Activa codigo de control 'SI'.
30 PRINT#8,CHR$(18);:REM Desactiva
codigo 'SI' mediante codigo de cont
rol 'DC2'.
40 PRINT#8,CHR$(27);"M";"Cuerpo 'EL
ITE': 12 caracteres por pulgada.":R
EM Modo 'ELITE'.
50 PRINT#8,CHR$(27);"P";"Cuerpo 'PI
CA': 10 caracteres por pulgada.":RE
M Desactiva modo 'ELITE' dejando la
 impresora en su estado normal (mod
o 'PICA').
60 PRINT#8,CHR$(27);"p";CHR$(1);"Cu
erpo 'PROPORCIONAL': espaciado prop
orcional.":REM Modo 'PROPORCIONAL'.
70 PRINT#8,CHR$(27);"p";CHR$(0);:RE
M Desactiva modo 'PROPORCIONAL'.
80 PRINT#8,CHR$(14);"Modo 'EXPANDID
O': 5 caracteres/pulgada.:rem Activ
a codigo de control 'SO'.
90 PRINT#8,CHR$(20);:REM Desactiva
codigo 'SO' mediante codigo de cont
rol 'DC4'.
100 PRINT#8,"Caracteres ";
110 PRINT#8,CHR$(27);"S";CHR$(1);"S
UB-Indices";:REM Modo 'SUBINDICES'.
120 PRINT#8,CHR$(27);"T";" y ";:REM
 Desactiva modo 'SUBINDICES' volvie
ndo a 'PICA'.
130 PRINT#8,CHR$(27);"S";CHR$(0);"S
UPER-Indices":REM Modo 'SUPERINDICE
S'.
140 PRINT#8,CHR$(27);"T";"Palabras
";:REM Desactiva 'SUPERINDICES' vol
viendo a 'PICA' para escribir "Pala
bras ".
150 PRINT#8,CHR$(27);"-";CHR$(1);"s
ubrayadas.":REM Modo 'SUBRAYADO'.
160 PRINT#8,CHR$(27);"-";CHR$(0);:
REM Desactiva modo 'SUBRAYADO'.
170 PRINT#8,CHR$(27);"E";"Caractere
s 'PICA' en 'NEGRITA' por doble pas
ada horizontal.":REM Modo 'NEGRITA'
.
180 PRINT#8,CHR$(27);"F";:REM Desac
tiva 'NEGRITA'.
190 PRINT#8,CHR$(27);"G";"Caractere
s 'PICA' en 'REPICADO' por doble pa
sada en vertical.":REM Modo 'REPICA
DO'.
200 PRINT#8,CHR$(27);"H";:REM Desac
tiva 'REPICADO'.
210 PRINT#8,STRING$(4,CHR$(10));REM
 Ordena cuatro saltos de linea.
220 FOR n=1 TO 3:PRINT#8,CHR$(27);"
0";"Espaciado entre lineas a 1/8 de
 pulgada.":NEXT:REM Espaciado a 1/8
".
230 PRINT#8,CHR$(27);CHR$(64):REM R
esetea la impresora
240 FOR n=1 TO 3:PRINT#8,CHR$(27);"
1";"Espaciado entre lineas a 7/72 d
e pulgada.":NEXT:REM Espaciado a 7/
72
250 PRINT#8,CHR$(27);CHR$(64)
260 FOR n=1 TO 3:PRINT#8,CHR$(27);"
2";"Espaciado entre lineas a 1/6 de
 pulgada.":NEXT:REM Espaciado a 1/6
"
270 PRINT#8,CHR$(27);CHR$(64)
280 FOR n=1 TO 3:PRINT#8,CHR$(27);"
3";CHR$(10);"Espaciado entre lineas
 a 10/216 de pulgada.":NEXT:REM Esp
aciado a n/216" (n=10).
290 PRINT#8,CHR$(27);CHR$(64)
300 FOR n=1 TO 3:PRINT#8,CHR$(27);"
A";CHR$(2);"Espaciado entre lineas
a 2/72 de pulgada.":NEXT:REM Espaci
ado a n/72" (n=2).
310 PRINT#8,CHR$(27);CHR$(64)
```

SALIDA DEL PROGRAMA 1

```
Cuerpo 'CONDENSADO': 17 caracteres
por pulgada.

Cuerpo 'ELITE': 12 caracteres por p
ulgada.

Cuerpo 'PICA': 10 caracteres por pu
lgada.

Cuerpo 'PROPORCIONAL': espaciado pr
oporcional.

Modo 'EXPANDIDO': 5 caracteres/pulg
ada.:rem Activa codigo de control '
SO'.
Caracteres SUB-Indices y SUPER-Indices
Palabras subrayadas.

Caracteres 'PICA' en 'NEGRITA' por
doble pasada horizontal.

Caracteres 'PICA' en 'REPICADO' por
 doble pasada en vertical.

Espaciado entre lineas a 1/8 de pul
gada.

Espaciado entre lineas a 1/8 de pul
gada.

Espaciado entre lineas a 1/8 de pul
gada.

Espaciado entre lineas a 7/72 de pu
lgada.

Espaciado entre lineas a 7/72 de pu
lgada.

Espaciado entre lineas a 7/72 de pu
lgada.

Espaciado entre lineas a 1/6 de pul
gada.

Espaciado entre lineas a 1/6 de pul
gada.

Espaciado entre lineas a 1/6 de pul
gada.

Espaciado entre lineas a 10/216 de
pulgada.
Espaciado entre lineas a 10/216 de
pulgada.
Espaciado entre lineas a 10/216 de
pulgada.

Espaciado entre lineas a 2/72 de pu
lgada.

Espaciado entre lineas a 2/72 de pu
lgada.

Espaciado entre lineas a 2/72 de pu
lgada.
```

Anote este nombre y recuerde esta imagen



SOFTWARE para AMSTRAD

- ***Si necesita uno de los mejores programas originales de gestión.***
  - Gestión integrada, facturación, stock, clientes.
- ***Si desea mecanizar su negocio.***
  - Clínicas veterinarias.
  - Vídeo clubs.
  - Administración de fincas.
  - Distribuidoras de cine.
- ***Si precisa un buen programa técnico.***
  - Cálculo de estructuras.
  - Mediciones y presupuestos.
  - Cálculo de vigas.
  - Estructuras espaciales.
  - Andamios.
  - Cálculos en hormigón.
- ***Si lo que quiere es una aplicación puntual.***
  - Declaración de la Renta.
  - Lotería primitiva.
  - Agenda multiuso.
- ***Si busca que le hagan un programa a la medida.***

ESPECIAL



SOFTWARE NEW LINE, S. A.
Gabinete de Informática
Zurbano, 4 – 28010 Madrid – Telfs. 410 40 98 - 410 47 63

SOLICITE INFORMACION MAS DETALLADA

Nombre y apellidos

Empresa

Dirección (C.P.)

Interesado en programas:

FICHEROS INDEXADOS Y MALLARD BASIC

Javier Barceló

En el primer número especial de MICROHOBBY AMSTRAD se trató el tema de los ficheros en disco para la serie de CPC. La aparición del PCW 8256, con un Basic distinto e incompatible con el de los CPC, entre otras particularidades incorpora un interesante módulo de programación, muy raro de encontrar en el Basic de ordenadores incluso muy superiores en precio: el módulo JETSAM para indexación de ficheros.

Los que leyeran el artículo anterior además de los que hayan hecho programas con ficheros, recordarán el inconveniente de los ficheros de acceso aleatorio. Cada registro se localizaba por medio de su número, por lo que si éste no era fácil de recordar, mediante una fórmula, o por que éste coincida, por ejemplo, con el número de cliente, etc... resultaba que para buscar un registro había que leer todos los anteriores, con lo que la principal facilidad que otorgaban estos ficheros quedaba anulada. Este problema es el origen de los ficheros indexados. Un fichero indexado es un fichero que posee un índice, al igual que un libro, de manera que la localización de un registro se hace a través de éste.

Un fichero indexado consta en realidad de dos ficheros. Uno donde se encuentran los datos introducidos, y otro donde el programa, a medida que se introducen los datos en el primero, va creando un índice de los registros. De esta manera, cuando se desee localizar un registro, basta con introducir el valor del campo clave —numérico o alfanumérico— y el propio programa lee el índice hasta localizar la clave, con la que está la posición que el registro ocupa en el fichero de datos, y después va al registro de datos.

Aunque el proceso pueda parecer lento, el ahorro de velocidad que así se consigue es muy considerable, dado que el fichero de índices no sólo es más corto que el de datos, sino que además se mantiene alfabéticamente ordenado según el valor de la clave. La clave puede ser cualquier campo de registro, y además no tiene por qué ser única. Es decir, que se puede mantener un fichero con un índice ordenado de varias maneras, especificando varias claves.

Naturalmente, el manejo de estos ficheros es un poco más complicado que el de los ficheros directos, dado que exige un mayor control del mismo y, por lo tanto, tiene más comandos, pero en grandes ficheros este modo de organización es difícilmente superable.

Módulo JETSAM

El Mallard Basic del PCW 8256 gestiona este tipo de ficheros a través de un módulo llamado JETSAM. Este módulo se encarga de gestionar —simultáneamente a la gestión normal del programa— el fichero de claves. Permite hasta ocho tipos de claves, que serán campos del registro con una longitud máxima de 32 caracteres por clave. Por el programa se selecciona el campo o campos que servirán de clave, su longitud y el orden de dichas claves. Si se utilizan varias claves, hay que especificar un orden de prioridad de las mismas. Esto es lo que se llama RANGO de la clave. Es decir, que existirán ocho rangos de claves como máximo, en cada uno de los cuales estarán clasificados todos los registros del fichero por distintos conceptos. Disponer de varias claves. Resulta imprescindible, por ejemplo, cuando en la primera clave seleccionada puedan existir valores repetidos en diversos registros. (Suponiendo un fichero que se ordene alfabéticamente, el primer apellido puede ser el primer rango de claves, y si hay repetidos, el segundo rango sería el segundo apellido, el tercero sería el nombre...).

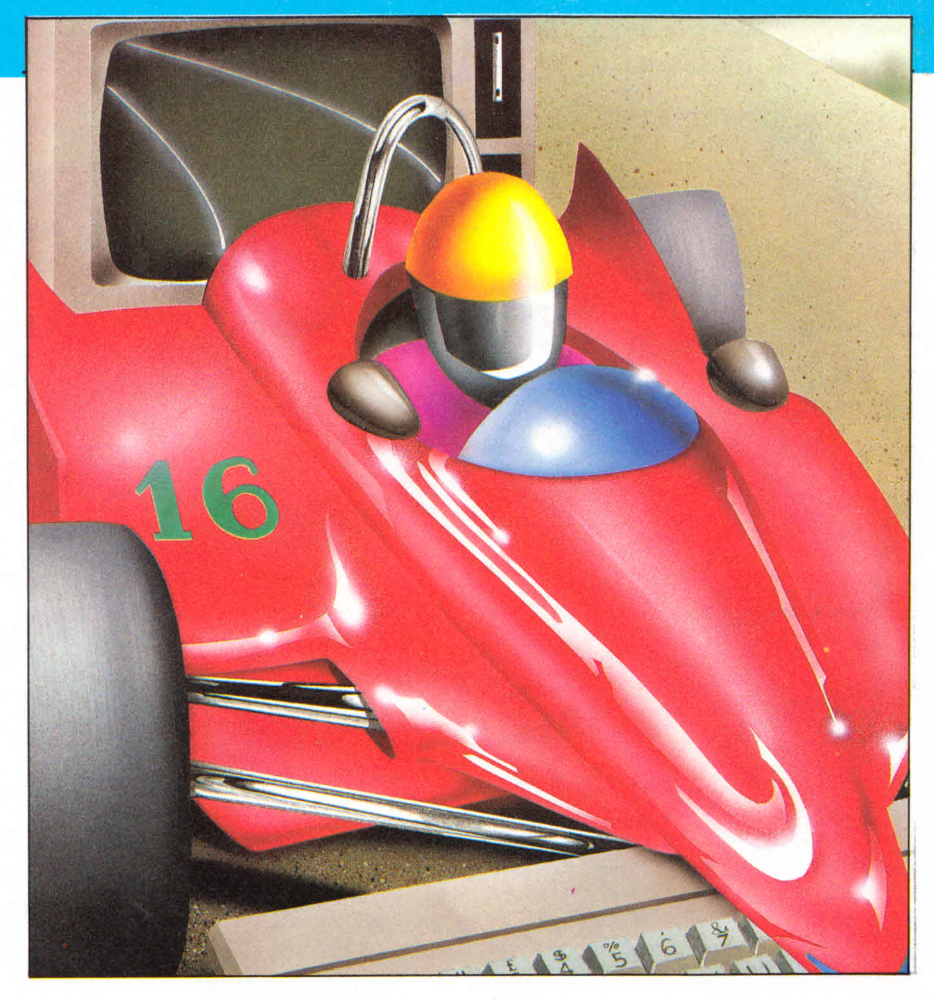
La manera de operar con los registros de este módulo es mediante una marca invisible que se llama puntero. El registro que se utiliza es aquél al que está señalando este puntero. Como el número de registro en estos ficheros es gestionado por el módulo JETSAM de manera automática, lo que hace el programa es **«ordenar»** a JETSAM que busque el registro con una clave determinada. Cuando JETSAM lo encuentra, señala el registro del fichero de datos al que corresponde esa clave con este puntero. Y entonces es cuando se consulta, modifica o cualquier otra operación que se desee hacer con él. Comprender que lo que se manejará para seleccionar cualquier registro es el puntero y no el número de registro, es fundamental para entender el manejo de diversas funciones que se explicarán a continuación.

Las operaciones con ficheros indexados se hace casi todas a través de funciones en vez de comandos. Esta es la principal diferencia con los demás tipos de ficheros. Una función la realiza y además da un resultado diferente, dependiendo del éxito o fracaso de dicha operación. Por esto, al utilizar una función ésta se coloca en el segundo término de una ecuación y en el primer término se coloca una variable. Al efectuarse la operación, la variable pasa a almacenar un valor que informa sobre el resultado de la operación. Es recomendable utilizar esta posibilidad asociada a una subrutina de detección de fallos, para tener la certeza de que la operación se ha efectuado correctamente. Existen distintos resultados según la función que se emplee, y en todos el valor 0 quiere decir que se ha efectuado la operación correctamente, mientras que los demás valores, que pueden diferir según la función, informan de distintas razones por las que ésta no se ha podido hacer con normalidad.

Para explicar las diferentes fases del funcionamiento de estos ficheros, se usará como referencia el programa ejemplo que aparece con este artículo. Este programa, muy sencillo, proporciona la clasificación de los equipos en la liga, introduciéndole primero los equipos y después los resultados de las jornadas. El fichero indexado es el fichero de equipos, dónde se actualizan los puntos y los goles cada vez que se introducen los resultados de una jornada. Los resultados de las jornadas van en un fichero directo muy sencillo, por lo que no se entrará en detalles de éste. La única particularidad que presenta este programa es que está hecho para doce equipos, por lo que si se desea utilizar con más, habrá que modificar en la línea 90, la variable EQUIPOS.

Creación del fichero

La primera diferencia que tienen estos ficheros con los demás es que éstos deben ser creados en el disco antes de ser utilizados. Pero antes de esto, cuando se quiera operar con ficheros indexados, lo primero que se hará es reservar sitio en la memoria para que JETSAM mantenga los índices del programa. Esto es necesario porque JETSAM mantiene desde que el fichero se abre hasta que se cierra la información sobre el índice en la memoria, para incrementar la velocidad del programa. Esto lo hace creando unas zonas reservadas en la memoria, llamadas buffers o tampones, que son bloques de memoria de 128 bytes. El número de tampones que se reservan no es fijo, se puede elegir teniendo en cuenta que a mayor cantidad de ellos, más rápidamente funcionarán las operaciones con el fichero, aunque naturalmente quede menos memoria disponible para el resto del programa. En casos





normales es suficiente con reservar seis buffers. La orden para hacer esto es:

BUFFERS 6

Esta orden habrá que utilizarla siempre al principio de todo programa que utilice ficheros indexados. En el programa ejemplo está en la línea 100.

La creación del fichero, propiamente dicho, se hace en la línea 150. Para esto, se usa la orden:

CREATE

La descripción de la orden es la siguiente: Después de CREATE, y sin coma, poner el número de fichero. Este es el número con el que nos referiremos a él en sucesivas instrucciones. Después poner una coma, entre comillas el nombre del fichero de datos, coma, nombre del fichero de claves, coma, dos (número necesario aunque inútil), coma, y la variable que contenga la longitud del fichero. (En el ejemplo, long.)

En cuanto a la longitud del fichero, hay que puntualizar una cosa. Cuando se esté haciendo el diseño de registro, la longitud total del mismo se verá incrementada en dos posiciones. Estas dos posiciones se ignorarán en todo momento, excepto al dar el valor de la longitud del registro, y son utilizadas internamente por el módulo JETSAM para codificar la relación con el fichero de claves.

Al crear el fichero éste queda abierto, por lo que si no se va a introducir ningún dato, es necesario cerrarlo. Al contrario que los ficheros secuenciales y aleatorios, para cerrar los ficheros indexados hay que especificar necesariamente el número del fichero que se cierra con la instrucción:

CLOSE No. de fichero.

El cierre del fichero es fundamental. Como se ha dicho antes, las actualizaciones en los índices se hacen en la memoria, y no se graban físicamente en el disco hasta que se cierra el fichero, por lo que un olvido, o un fallo de corriente, y otra incidencia que impida el cierre correcto, haría que dicho fichero no se grabase, y el fichero de datos quedase inutilizado. Es lo que se llama fichero inconsistente. Para mayor seguridad, si el fichero debe estar abierto mucho tiempo, se puede usar el comando CONSOLIDATE. Este graba el contenido de los tampones de claves en el disco, dejándolo en el mismo estado en que quedaría después de cerrarlo, pero abierto. Equivale a hacer CLOSE, y después OPEN.

Mantenimiento del fichero

Para abrir un fichero indexo, naturalmente ya creado, se utiliza la instrucción:

OPEN «K»

En esta instrucción, que se puede ver en las líneas 300, 530 y 1020, después de la K entre comillas, indica que el fichero es de acceso por claves, hay que poner el número del fichero, que no tiene por qué coincidir con el que se puso al crearlo. El número que se da al abrirlo es el que luego servirá para referirse a este fichero. Después de éste, hay que poner entre comillas el nombre del fichero de datos y el de claves, el número 2 y la variable que contenga la longitud del registro, todos ellos separados por comas. La longitud del registro se puede omitir si ésta es de 128 caracteres, pero hay que ponerla en los restantes casos.

Aun hay otra instrucción necesaria antes de poder operar con estos ficheros. Esta sirve para definir el diseño del registro, y es:

FIELD número de fichero, longitud AS nombre de campo.

Si se ven las líneas 310, 550 y 1030 se observa que después de la instrucción viene el número del fichero al que se refiere, y posteriormente, separados por comas, los campos de los que consta el registro con su longitud. Se pone primero la longitud del primer campo, la palabra (AS) y su nombre, y separados por comas los demás campos que tengan el registro. No hay que considerar los dos caracteres que reserva JETSAM para su uso, luego la suma de la longitud de los campos que aparezcan en este comando, debe sumarse la longitud total del registro menos dos. En este punto hay que aclarar que a la hora de introducir datos en las variables que aparecen en esta instrucción, NO se puede hacer directamente. Hay que utilizar las instrucciones LSET, RSET y MID$. Si se ve la línea 750, no se lograría el mismo efecto haciendo:

nombre$ = nom1$

sino que se crearía una variable llamada nombre$ que no tiene nada que ver con la definida en la instrucción FIELD. Hay, pues, que poner antes LSET, RSET o MID$ ANTES de la variable nombre$ LSET coge de la variable de la segunda parte de la igualdad empezando por la izquierda, el número de caracteres necesario para completar la longitud dada a la variable de la primera parte en la instrucción FIELD. RSET hace lo mismo, pero empezando por la derecha. Y MID$ permite elegir una zona intermedia de dicha variable.

Explicadas las primeras instrucciones, hay que pasar a explicar las funciones de las que dispone JETSAM para gestionar estos ficheros. Antes que nada, se les llama funciones porque dan un resultado. Es decir, que al escribir, leer, buscar, etc... podemos saber si el resultado es correcto o no mediante el resultado de la función. La línea 430 del ejemplo, almacena en la variable RDO el resultado de una función (que se explicará más tarde) y va a la subrutina de confirmación de resultados, a partir de la línea 1360, para comprobar cuál ha sido el resultado de la operación realizada, y lo presenta en pantalla. Aunque una vez

depurado su programa y comprobado que funciona correctamente, esto no sea en absoluto necesario, mientras se construye y se prueba el programa resulta de suma utilidad para darse cuenta de algunos fallos difíciles de notar de otro modo.

Una vez explicado esto, pasemos revista a las diversas funciones que incluye el módulo JETSAM.

ADDREC escribe los datos que estén almacenados en las variables de la instrucción FIELD en el registro siguiente al último que haya en el fichero, y el puntero queda apuntándole. Además añade la clave de dicho registro en el fichero de claves. Como se ve en la línea 430 del ejemplo, la forma es:

Variable=ADDREC(No. Fichero,2,rango, clave)

En el campo (Variable) se almacena el resultado de la operación que añade el registro al fichero, y además toma el campo que aparezca en *(clave)* y los sitúa en el rango indicado actualizando dicho rango de claves.

Pero como se ha dicho, se puede tener clasificado un fichero por distintas claves. En ese caso, después de esta instrucción hay que poner esta otra:

Variable=ADDKEY(No. Fichero,0,rango,clave,No. de Registro)

El funcionamiento de ésta es similar al de la función anterior, pero hay que añadir una cosa: el número de registro de dicha clave. Para esto, habrá que averiguar el No. de registro que acaba de utilizar ADDREC. (dado que hay que incorporar una nueva clave al registro que ha sido grabado anteriormente.) Esto se hace a través de la siguiente función:

Variable= FETCHREC (No. de fichero)

En esta variable tomará el valor del registro al que esté señalando el puntero, pudiendo incluirlo después en otras funciones. Esto es lo que hacen las líneas 450 y 460 del ejemplo.

También se puede necesitar saber el rango de la última clave introducida. Para esto se utiliza:

Varible=FETCHRANK (No. de fichero)

Donde variable tomará el valor del último rango que se haya utilizado. En caso de necesitar el contenido de la última clave utilizada, se usa:

Variable$= FETCHKEY$(No. de Fichero)

Que como en anteriores funciones, almacenará en variables$ el contenido de la última clave utilizada.

Para borrar una clave del fichero, hay que tomar alguna precaución. Si el registro al que corresponde esa clave no tiene además otra clave, es decir, si ésta es única, al borrarla es como si borrásemos el registro de datos, dado que no tenemos manera de acceder a él. Por esto, si sólo se desea cambiar una clave por otra, es recomendable primero añadir la clave nueva (ADDKEY) y luego borrar la actual. Esta se borra a través de la función:

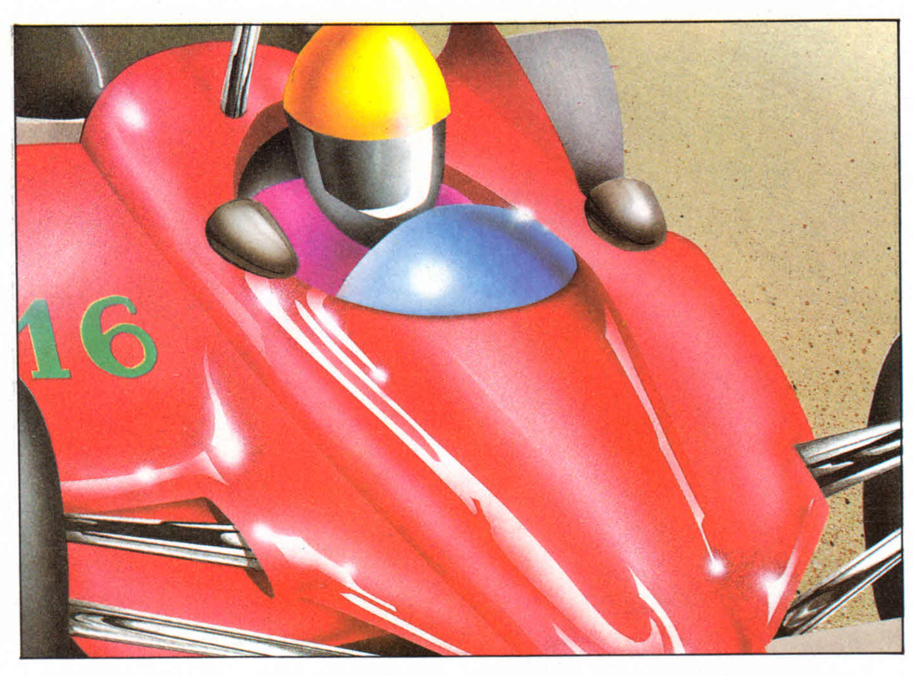
Variable= DELKEY *(No.Fichero,0,posición)*

La posición es optativa. Si no se pone, borrará la posición a la que señale el puntero en ese momento, por lo que si no es esa la que se desea borrar, habrá que situar el puntero previamente en la clave deseada, o bien investigar la posición de la clave que se quiera borrar. Ambas cosas se hacen a través de las funciones de búsqueda, que se ven posteriormente.

Se puede optar entre admitir claves repetidas o no. Si se quiere impedir que el programa admita claves repetidas, hay que usar la función RANKSPEC de la siguiente manera:

Variable=RANKSPEC *(No.Fichero,No. Clave,0)*

Esta función, no aparece explicada en el manual de instrucciones más que de una manera superficial.



Búsqueda de registros

Dado que el número de registro aquí sirve para bien poco, el módulo JETSAM proporciona varias funciones que se encargan de buscar el registro que se desea. El resultado de estas funciones permite saber si se ha encontrado otro registro con la misma clave, una clave distinta dentro del mismo rango, una clave nueva en distinto rango, o que no se ha encontrado dicha clave. Antes de empezar con las instrucciones, hay que entender que sólo hay un fichero de claves, aunque se disponga de varios rangos de indexación. A partir de la última clave de un rango, empieza la primera clave del siguiente rango, por lo que la búsqueda en un rango puede dar positiva en otro rango distinto. Para este caso, el resultado de la función NO es el mismo, por lo que en todo momento se puede saber dónde se ha encontrado, y si interesa el registro encontrado o no.

SEEKKEY busca en el fichero el primer registro cuya clave sea igual a la dada, dentro del rango dado. En el ejemplo, está en línea 1340. La forma de esta función es:

varible= SEEKKY(No. Fichero,0,rango,clave)

Si hay más registros con idéntica clave, y se quiere seguir con la búsqueda, se utilizará SEEKNEXT, que sigue recorriendo el fichero buscando la misma clave en el mismo rango anteriormente dados. En el ejemplo está en la línea 1190 y su forma es:

variable= SEEKNEXT (No. Fichero,0,posición de búsqueda)

La posición de búsqueda es opcional, de manera que si no se da buscará en el mismo rango a partir de la posición a la que señale el puntero, y en caso contrario a partir de la posición que se indique.

La función SEEKPREV es idéntica a la anterior, pero busca la clave anterior a la indicada. Su sintaxis es también idéntica a la anterior.

SEEKSET es útil en caso de que el fichero de índices contenga claves repetidas. Su sintaxis es:

Variable=SEEKSET (No.Fichero,0,posición de búsqueda)

Si hay claves repetidas, esta función se salta las claves iguales a la actual, hasta la primera clave distinta de éstas.

Lectura y escritura en el fichero

Hasta ahora se ha hablado principalmente del puntero que señala al registro. NO se ha hablado de cómo extraer información del fichero, ni cómo reescribir la misma en un registro que ya existía. Hay que recordar que ADDREC añade un registro después del último, pero no sirve para modificar datos en re-

gistros intermedios. Pues bien, si se ha entendido lo anterior, esto está **«chupao».** Simplemente se coloca el puntero señalando al registro deseado, y luego si se quiere leer el registro, se utiliza la instrucción:

GET No. de fichero

o si se quiere escribir en ese registro, (una vez dados sus valores a los campos del registro mediante LSET, etc...) se utiliza la instrucción:

PUT No. de fichero

Como se ve, no hay que poner el número del registro que se desea leer o escribir. Es el puntero el que se encarga de indicar cuál es el registro que se va a leer o escribir. Por lo tanto, sólo hay que ocuparse de que éste señale al registro que se desea. En el ejemplo, las líneas 600, 870, 940 y 1120 realizan esta función.

Conclusiones

Las peculiares características de este tipo de ficheros, lo hacen especialmente adecuadas para programas de gestión. Programas como controles de stock, contabilidad, presupuestos, etc... sin disponer de ficheros indexados serían muy complejos, y excesivamente lentos. El funcionamiento de este módulo es muy bueno, aunque un tanto complicado. Pero el problema es entender su funcionamiento, no hacer el programa. Una vez entendido, cosa que se logra con un poco de práctica, nuestro horizonte de programación se amplía considerablemente. Desgraciadamente sólo está al alcance del PCW y no de los CPC con disco. Para éstos, se puede simular la indexación, realizando a través del programa algunas de las funciones que hacen estos ficheros. El mayor problema será actualizar el fichero de índices cada vez que se modifique el de datos. Es una tarea de programación complicada, pero efectiva.

Por último un consejo. Para realizar un buen programa con estos ficheros, es necesario tener las ideas muy claras. Pensar siempre en el camino más corto, y éste suele ser utilizar la mayoría de las funciones que JETSAM nos proporciona. Si no se es remiso a la hora de utilizarlas, nuestro programa será un poco más largo, pero mucho más rápido y eficaz. Y para pensar el programa, lápiz y papel. Antes de meterse con el ordenador, definir claramente lo que se necesita y cómo se quiere hacer. Si esto es necesario en practicamente todos los programas, si manejan este tipo de ficheros es imprescindible.

Estructura del programa de ejemplo

Para demostrar algunas de las funciones que se han explicado anteriormente, obsérvese la estructura del programa del ejemplo:

Línea **100:** Se establece el número de tampones que se van a reservar.

Línea **120:** Se busca en el disco si existen el fichero de datos y el de claves.

Línea **150:** Si no existen los ficheros, se crean. Se ha optado por darles el mismo nombre con distinta extensión por cuestión de costumbre, pero ambos nombres pueden ser diferentes en todo.

Línea **160:** Se cierra el fichero. Aunque para introducir datos no haría falta cerrarlo, como sólo se va a crear el fichero una vez pero habrá que abrirlo cada vez que se necesite, en este caso es necesario cerrrarlo.

Línea **300:** Se abre el fichero. La longitud del registro está almacenada en la variable Long, en la línea 100.

Línea **310:** Se establecen la longitud y el nombre de los campos que forman el registro. La suma de la longitud de todos los campos es 37, que sumados a los 2 caracteres que necesita JETSAM para gestionar el fichero, da los 39 de la longitud del registro.

Línea **400:** Se almacenan los nuevos valores en las variables mediante LSET. Los números se almacenan empaquetados mediante MKI$.

Línea **430:** Se graba el registro, y se verifica el resultado de la operación, mediante RDO. El primer rango de claves se establece con los puntos de cada equipo.

Línea **450:** Se averigua el número del registro al que señala el puntero mediante FETCHREC.

Línea **460:** Utilizando el dato anterior, se establece un nuevo rango usando como campo de claves el nombre del equipo.

Línea **810:** Se va a la subrutina de búsqueda, para situar el puntero en el registro correspondiente al primer equipo.

Línea **870:** Una vez modificados los datos del registro, se graba.

Línea **880:** Se va a la subrutina de búsqueda, para situar el puntero en el registro correspondiente al segundo equipo.

Línea **940:** Igual que la línea 870.



Línea **950:** Se consolidan el fichero de datos y el de claves, quedando listos para introducir más resultados. No es necesario hacerlo, pero puede evitar fallos, aunque haga el programa un poco más lento.

Línea **1100:** Buscamos la primera clave del primer rango. El puntero queda señalando a esta clave.

Línea **1120:** Se lee el registro al que señala el puntero.

Línea **1190:** Pone el puntero señalando la siguiente clave, y repite el bucle de lectura.

Línea **1330:** Principio de la subrutina de búsqueda. Sitúa el puntero señalando la primera clave del rango 1.

Línea **1340:** A partir de la posición anterior, busca en este rango el dato pedido, que está almacenado en la variable NOMBRE$.

Línea **1370:** Comienza la subrutina que confirma el resultado de las operaciones. En principio, presentar esto en pantalla es útil para saber si funciona correctamente, aunque luego se pueda suprimir.

Nota sobre campos empaquetados

La función MKI$ transforma un número entero en una cadena compacta con una longitud de 2 bytes. Así se optimiza su almacenamiento en el disco, ahorrando espacio. Pero un número **«empaquetado»** no puede ser presentado en pantalla, ni se puede operar con él. Hay que reconvertirlo, mediante la función CVI.

FICHERO INDEXADO EQUIPOS. DAT

Variable	Descripción	Long.	Tipo
*Nombre$	Nombre del Equipo	15	Caracteres
PGC$	Partidos ganados casa	2	Empaquetado
PGF$	Partidos ganados fuera	2	Idem.
PEC$	Partidos empatados casa	2	Idem.
PEF$	Partidos empatados fuera	2	Idem.
PPC$	Partidos perdidos casa	2	Idem.
PPF$	Partidos perdidos fuera	2	Idem.
GFC$	Goles a favor en casa	2	Idem.
GCC$	Goles en contra en casa	2	Idem.
GFF$	Goles a favor fuera	2	Idem.
GCF$	Goles en contra fuera	2	Idem
*PTS$	Puntos totales	2	Idem
		37	caracteres

* Campos clave.

```
10 '**************************
20 '    MICROHOBBY  AMSTRAD
30 '    Ficheros Indexados
40 '     AMSTRAD PCW 8256
50 '          F.J.B.T.
60 '**************************
70 '**** INICIALIZACION.
80 cls$=CHR$(27)+"E"+CHR$(27)+"H"
90 Equipos=12:NR=INT(equipos/2)
100 BUFFERS 6:long=39:DIM cla$(equi
pos),p(equipos)
110 PRINT cls$:PRINT:PRINT:PRINT:
120 a$=FIND$("equipos.dat"):b$=FIND
$("equipos.key")
130 IF a$<>"" AND b$<>"" THEN 170
140 '**** CREACION DE LOS FICHEROS
150 CREATE 1,"EQUIPOS.DAT","EQUIPOS
.KEY",2,long
160 CLOSE 1
170 '**** MENU PRINCIPAL
180 PRINT cls$
190 PRINT:PRINT:PRINT:PRINT:PRINT:P
RINT TAB(35) "MENU GENERAL"
200 PRINT TAB (35)"============"
210 PRINT:PRINT TAB (30) " 1.- ALTA
S de Equipos."
220 PRINT:PRINT TAB (30) " 2.- Intr
oduccion de Resultados."
230 PRINT:PRINT TAB (30) " 3.- Clas
ificacion."
240 PRINT:PRINT TAB (30) " 4.- Fin
de Programa."
250 PRINT:PRINT:PRINT TAB (25);:INP
UT "Seleccione Opcion.:",Op
260 IF op<1 OR op>4 THEN 180
270 ON op GOTO 280,520,1010,1300
280 '**** OPCION 1
290 '**** Apertura Fichero Indexado
.
300 OPEN "K",1,"EQUIPOS.DAT","EQUIP
OS.KEY",2,long
310 FIELD 1,15 AS Nombre$,2 AS PGC$
,2 AS PGF$,2 AS PEC$,2 AS PEF$,2 AS
 PPC$,2 AS PPF$,2 AS GFC$,2 AS GCC$
,2 AS GFF$,2 AS GCF$,2 AS Pts$
320 PRINT cls$
330 PRINT:PRINT:PRINT:PRINT TAB (30
) "ALTAS DE EQUIPOS"
340 PRINT TAB (30) "---------------
-"
350 PRINT:PRINT:PRINT TAB (25);:INP
UT "Nombre.   15 C.: ";nom$:nom$=UPP
ER$(nom$)
360 IF LEN(nom$)>12 THEN PRINT:PRIN
T TAB (25) " Longitud Excesiva": GO
TO 480
370 LSET nombre$=nom$
380 GOSUB 1320: IF busq=0 THEN PRIN
T:PRINT TAB (25) "Registro existent
e": GOTO 480
390 Da=0
400 LSET pgc$=MKI$(da):LSET pgf$=MK
I$(da):LSET pec$=MKI$(da):LSET pef$
=MKI$(da):
410 LSET ppc$=MKI$(da):LSET ppf$=MK
I$(da):LSET gfc$=MKI$(da):LSET gcc$
=MKI$(da):
420 LSET gff$=MKI$(da):LSET gcf$=MK
I$(da):LSET pts$=MKI$(da)
430 rdo=ADDREC(1,2,0,pts$)
440 GOSUB 1360
450 numreg=FETCHREC(1)
460 rdo=ADDKEY(1,0,1,nombre$,numreg
)
470 GOSUB 1360
480 PRINT:PRINT:PRINT TAB (25);:INP
UT "¿Desea grabar otro registro? (S
/N).: ",re$:re$=UPPER$(re$)
490 IF re$="S" THEN 320
500 CLOSE 1:GOTO 170
510 '*******************************
520 '**** OPCION 2
530 OPEN "K",1,"EQUIPOS.DAT","EQUIP
OS.KEY",2,long
540 OPEN "R",2,"JORNADAS.DAT",35
550 FIELD 1,15 AS Nombre$,2 AS PGC$
,2 AS PGF$,2 AS PEC$,2 AS PEF$,2 AS
 PPC$,2 AS PPF$,2 AS GFC$,2 AS GCC$
,2 AS GFF$,2 AS GCF$,2 AS Pts$
560 FIELD 2,1 AS c$,15 AS eq1$,2 AS
 g1$,15 AS eq2$,2 AS g2$
570 PRINT cls$
580 PRINT:PRINT:PRINT:PRINT TAB (30
) "INTRODUCCION DE RESULTADOS"
590 PRINT:PRINT:PRINT TAB (25);:INP
UT "N! de Jornada..: ",jor
600 GET 2,(1+((jor-1)*nr))
610 IF c$="*" THEN PRINT TAB (20);:
INPUT "Jornada con datos. ¿Desea re
introducirlos? (S/N).:",re$:re$=UPP
ER$(re$)
620 IF re$="N" THEN 570
630 FOR x=1 TO nr
640 PRINT cls$
650 PRINT:PRINT:PRINT:PRINT TAB(30)
 "INTRODUCCION DE RESULTADOS.  Jorn
ada N! ";jor
660 PRINT TAB (30) "---------------
-----------------------------"
670 PRINT:PRINT:PRINT TAB(25);: INP
UT "Equipo de Casa. 15 C.:", nom1$:
nom1$=UPPER$(nom1$)
680 PRINT:PRINT TAB (25);: INPUT "G
oles................:",go1
690 PRINT:PRINT:PRINT TAB(25);: INP
UT "Equipo Visitante 15C.:",nom2$:n
om2$=UPPER$(nom2$)
700 PRINT:PRINT TAB (25);: INPUT "G
oles................:",go2
710 PRINT:PRINT:PRINT:PRINT TAB(30)
;: INPUT "¿CONFORME? (S/N).: ",re$:
re$=UPPER$(re$)
720 IF re$="N" THEN 640
730 '**** Actualizar Fichero Direct
o
740 LSET c$="*"
750 LSET nombre$=nom1$:GOSUB 1320:
IF busq<>0 THEN 990
760 LSET nombre$=nom2$:GOSUB 1320:
IF busq<>0 THEN 990
770 LSET eq1$=nom1$:LSET g1$=MKI$(g
o1)
780 LSET eq2$=nom2$:LSET g2$=MKI$(g
o2)
790 PUT 2,(x+((jor-1)*nr))
800 '**** Actualizar Fichero Indexa
do
810 LSET nombre$=nom1$:GOSUB 1320:G
ET 1
820 IF go1>go2 THEN pgc=CVI(pgc$)+1
:LSET pgc$=MKI$(pgc):pts=CVI(pts$)+
2:LSET pts$=MKI$(pts)
830 IF go1=go2 THEN pec=CVI(pec$)+1
:LSET pec$=MKI$(pec):pts=CVI(pts$)+
1:LSET pts$=MKI$(pts)
840 IF go1<go2 THEN ppc=CVI(ppc$)+1
:LSET ppc$=MKI$(ppc)
850 LSET gfc$=MKI$(CVI(gfc$)+go1)
860 LSET gcc$=MKI$(CVI(gcc$)+go2)
870 PUT 1
880 LSET nombre$=nom2$:GOSUB 1320:G
ET 1
890 IF go1<go2 THEN pgf=CVI(pgf$)+1
:LSET pgf$=MKI$(pgf):pts=CVI(pts$)+
2:LSET pts$=MKI$(pts)
900 IF go1=go2 THEN pef=CVI(pef$)+1
:LSET pef$=MKI$(pef):pts=CVI(pts$)+
1:LSET pts$=MKI$(pts)
910 IF go1>go2 THEN ppf=CVI(ppf$)+1
:LSET ppf$=MKI$(ppf)
920 LSET gff$=MKI$(CVI(gff$)+go2)
930 LSET gcf$=MKI$(CVI(gcf$)+go1)
940 PUT 1
950 rdo=CONSOLIDATE(1)
960 NEXT
970 CLOSE 2:CLOSE 1:
980 GOTO 170
990 PRINT:PRINT:PRINT TAB (25) "Err
or en nombre de Equipo.":GOTO 640
1000 '******************************
1010 '**** OPCION 3
1020 OPEN "K",1,"EQUIPOS.DAT","EQUI
POS.KEY",2,long
1030 FIELD 1,15 AS Nombre$,2 AS PGC
$,2 AS PGF$,2 AS PEC$,2 AS PEF$,2 A
S PPC$,2 AS PPF$,2 AS GFC$,2 AS GCC
$,2 AS GFF$,2 AS GCF$,2 AS Pts$
1040 PRINT cls$
1050 PRINT:PRINT:PRINT TAB (30) "C
L A S I F I C A C I O N"
1060 PRINT TAB (30) "==============
==========="
1070 PRINT "EQUIPO                JUGA
 DOS  GANADOS  EMPATADOS  PERDIDOS
 GOL.FAVOR  GOL.CONTRA     PUNTOS"
1080 PRINT "C=Casa.  F=Fuera      C
 F     C   F   C      F     C     F
  C   F        C    F"
1090 PRINT STRING$(90,"-")
1100 rdo=SEEKRANK(1,0,0)
1110 FOR x=1 TO equipos
1120 GET 1
1130 pgc=CVI(pgc$):pec=CVI(pec$):pp
c=CVI(ppc$):pgf=CVI(pgf$):pef=CVI(p
ef$):ppf=CVI(ppf$)
1140 gfc=CVI(gfc$):gcc=CVI(gcc$):gf
f=CVI(gff$):gcf=CVI(gcf$):pts=CVI(p
ts$)
1150 pjc=pgc+pec+ppc:pjf=pgf+pef+pp
f
1160 cla$(x)= nombre$+"   "+STR$(pjc
)+"   "+STR$(pjf)+"    "+STR$(pgc)+"
 "+STR$(pgf)+"    "+STR$(pec)+"    "+
STR$(pef)+"     "
1170 cla$(x)=cla$(x)+STR$(ppc)+"
"+STR$(ppf)+"     "+STR$(gfc)+"   "+S
TR$(gff)+"     "+STR$(gcc)+"      "+S
TR$(gcf)+"      "+STR$(pts)
1180 p(x)=pts
1190 rdo=SEEKNEXT(1,0)
1200 NEXT
1210 FOR x=1 TO equipos
1220 FOR y=x+1 TO equipos
1230 IF p(y)<p(x) THEN p2=p(x):b$=c
la$(x):p(x)=p(y):cla$(x)=cla$(y):p(
y)=p2:cla$(y)=b$
1240 NEXT y
1250 NEXT x
1260 FOR x=equipos TO 1 STEP -1
1270 PRINT cla$(x):NEXT x
1280 PRINT:PRINT:PRINT:PRINT TAB (3
4);: INPUT "Pulse INTRO.";in$
1290 CLOSE 1:GOTO 180
1300 '**** OPCION 4
1310 CLOSE 1:CLOSE 2:END
1320 '**** SUBRUTINA DE BUSQUEDA.
1330 inic=SEEKRANK(1,0,1):PRINT "In
icializacion.: ";inic
1340 busq=SEEKKEY(1,0,1,nombre$):PR
INT "Busqueda........: ";busq
1350 RETURN
1360 '**** CONFIRMACION DE RESULTAD
OS.
1370 IF rdo=0 THEN PRINT:PRINT TAB(
25)" Operacion Correcta."
1380 IF rdo=101 THEN PRINT:PRINT TA
B(25)" Sobrepasado final de claves.
"
1390 IF rdo=102 THEN PRINT:PRINT TA
B(25)" Sobrepasado final de rangos.
"
1400 IF rdo=103 THEN PRINT:PRINT TA
B(25)" Sobrepasada ultima clave."
1410 IF rdo=105 THEN PRINT:PRINT TA
B(25)" Clave no encontrada."
1420 IF rdo=111 THEN PRINT:PRINT TA
B(25)" Posicion de registro indefin
ida."
1430 IF rdo>129 THEN PRINT:PRINT TA
B(25)"Error de bloqueo."
1440 RETURN
```



VIERNES 13

CASSETTE
2.500 pts.
DISCO
4.300 pts.





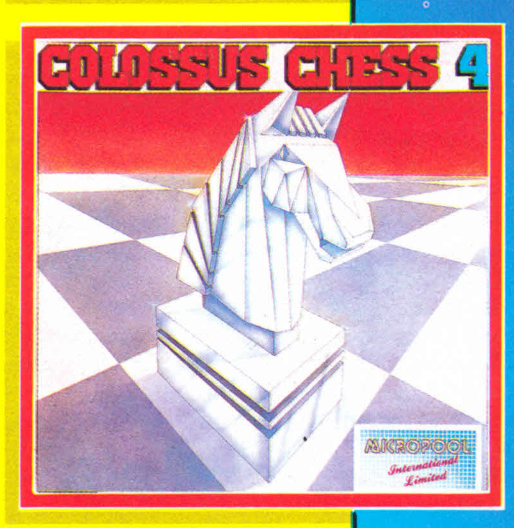
A VIEW TO A KILL

CASSETTE
2.500 pts.
DISCO
4.500 pts.



DISCO
4.300 pts.





COLOSSUS
CHESS 4

CASSETTE
2.600 pts.
DISCO
3.900 pts.



RECORTA Y ENVIA ESTE CUPON A: SERMA, C/. BRAVO MURILLO, N.º 377-3.º A., 28020 MADRID. TELEFONOS: 733 73 11 - 733 74 64

CODENAME MAT II ☐ VIERNES 13 (cassette) ☐ (disco) ☐ A VIEW TO A KILL (cassette) ☐ (disco) ☐ COLOSSUS 4 CHESS (cassette) ☐ (disco) ☐

NOMBRE Y APELLIDOS: ____________________

DIRECCION: ____________________

POBLACION: ____________________ PROVINCIA: ____________________

CODIGO POSTAL: ________ FORMA DE PAGO: ENVIO TALON BANCARIO ☐ CONTRA REEMBOLSO ☐

SISTEMAS OPERATIVOS

Autor: E. Riego

La CPU controla todos los componentes (recursos) del ordenador. Sin ello, el microprocesador estaría aislado del mundo exterior, y no serviría, por tanto, para nada. Para controlar los recursos físicos, basta mandar a sus interfaces (circuitos electrónicos intermedios que permiten la comunicación) las secuencias de bytes necesarias para lograr los efectos deseados.



Cada recurso puede realizar diferentes tareas y cada una posee una codificación, conocida por la CPU. Por ejemplo para que el programa sea capaz de leer un fichero de datos de un disco tiene que ordenar al controlador del camino que posicione la cabeza lectora sobre la cara, pista y sector adecuados (que el programa debe obtener de algún modo), y una vez allí leer los bytes que se necesiten. Esta operación se realiza muchas veces a lo largo de una sesión de trabajo con un ordenador. Para la acción anterior (y para muchas más) sería conveniente tener una rutina en código máquina a la que llamar cada vez que se necesitase.

El sistema operativo es un conjunto de programas o rutinas que se ocupan de gestionar los recursos y coordinar los sucesos que se producen en el ordenador. Proveen al programador de la mayoría de la herramientas básicas para realizar programas sin tener que repetir en cada uno las mismas rutinas,

Inicialmente los ordenadores carecían de sistema operativo, poseían una serie de clavijas que enchufadas en diferentes conectores producían las secuencias de introduciones necesarias para realizar las operaciones que especificaba el código de un programa. Posteriormente se descubre la actividad de tener sistemas operativos (formados por las rutinas de entrada y salida más básicas) dentro del ordenador en ROM, accesibles desde cualquier programa, sin tener que codificar el mismo programa por sí mismo.

Gracias a este cambio las compañías de software ahorraron mucho tiempo en el desarrollo de productos nuevos, pero seguía sin ser una solución totalmente satisfactoria ya que ataba a un ordenador a un sistema particular, que generalmente sólo utilizaba el mismo: los ordenadores seguían estando aislados.

De aquí surgen los sistemas operativos basados en disco, que poseen en ROM el código mínimo para cargar cierta parte de la información contenida en un disco y ejecutarla. Así, el mismo ordenador puede usar diferentes sistemas operativos leyéndolos de disco, sin cambiar para nada la estructura interna del mismo.

Gracias a estos sistemas operativos, un mismo programa puede funcionar en muchas máquinas diferentes que corran el mismo sistema operativo, sin que las casas creadoras hayan tenido que hacerle ningún cambio. Tenemos sistemas portables.

La portabilidad exige una estandarización en hardware, sobre todo en cuanto al microprocesador utilizado; deben ser iguales o compatibles en cuanto a su código máquina. Aun así no se puede asegurar que dos máquinas diferentes corriendo el mismo sistema operativo puedan usar el mismo programa sin haber cambios. Las razones de esta falta de compatibilidad se deben a varios factores, los fundamentales son que las configuraciones físicas de las máquinas no son exactamente las mismas o que el sistema operativo del ordenador de una de las máquinas incorpora una mejora añadida por el fabricante para aumentar las prestaciones del mismo, y ésta no es contemplada por la otra máquina (por ejemplo los diferentes formatos de disco encontrados en cda máquina que funciona bajo control de CP/M).

Los ordenadores domésticos baratos suelen disponer del lenguaje Basic en ROM, con un sistema operativo mezclado con el mismo código Basic (podría decirse que el Basic incluye dentro de su código las rutinas usuales de un sistema operativo), diseñado para aprovechar al máximo las posibilidades de la máquina y que, por tanto, sería muy costoso de llevar a otra.

Sistemas operativos actuales

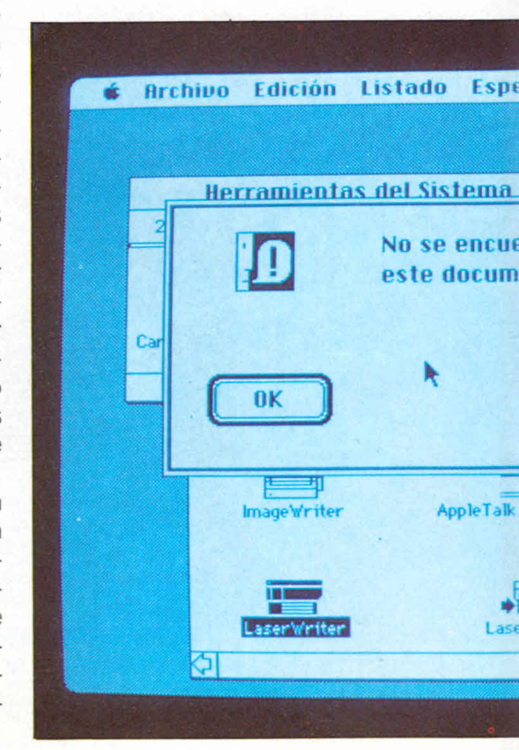
CP/M 80 fue el primer sistema operativo basado en disco para microordenadores basados en los microprocesadores Z80, 8080 y 8085, y el primero de una larga serie de sistemas que han aparecido después, a menudo utilizando características introducidas por él.

Debido a sus exigencias hardware sólo es capaz de direccionar 64 Kbytes de memoria (aunque la versión 3.0 o CPM/80+direcciona 128 mediante la técnica de conmutación de bancos de memoria), por lo que es muy compacto y sólo 8 Kb de memoria central, conteniendo las rutinas esenciales y 6 comandos residentes de uso general: DIR, ERA, REN, TYPE, SAVE, USER, que sirven para obtener el listado del directorio, borrar un fichero de disco, cambiarle el nombre, visualizarlo en pantalla, grabar en disco alguna zona de memoria y asignar un número de usuario, cada uno con diferentes ficheros asociados, accesibles sólo desde el usuario que los creó.

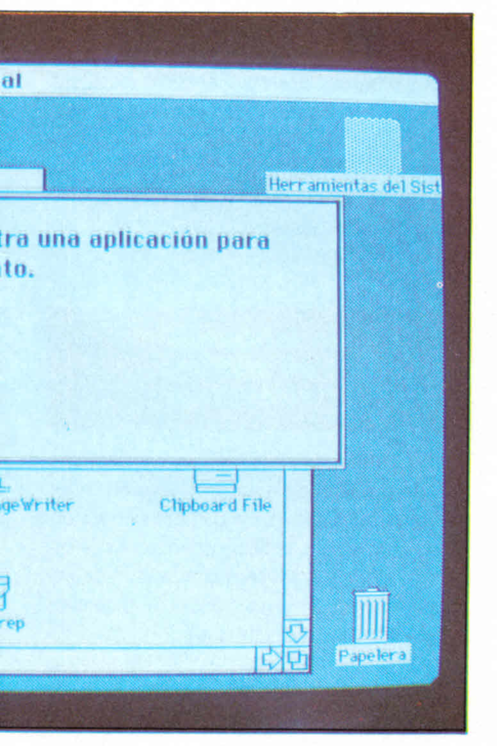
Estos comandos son resistentes porque permanecen en la memoria todo el tiempo, cualquier otra utilidad del sistema operativo o programa ejecutable permanece en disco como fichero. Cada fichero tiene asociado un nombre de 8 letras seguido opcionalmente de un punto y una extensión de 3 letras más. Los ficheros ejecutables tienen extensión COM.

Las unidades poseen los nombres A:, B:, C:, D:,... y una referencia completa a un fichero incluye la unidad de disco en que permanece (es opcional); más el nombre y extensión: A:FICHERO.DAT es un fichero que está en el disco A: con nombre FICHERO y extensión DAT.

Al cargar el sistema, aparece el indicador A+ (prompt) que nos informa de qué siste-



ma espera órdenes, manteniendo la unidad de disco A: como disco activo o por defecto, en la que se realiza cualquier operación que no lo especifique. Para cambiar, B: seguido de RETURN establece B: como disco defecto, apareciendo el indicador B + .



Cualquier cosa que se teclee seguida de RETURN es interpretada como una orden y el sistema intenta ejecutarla, mirando primero si es un comando interno (y ejecutándolo), buscando en el disco que se especifica o en el disco por defecto un fichero de tipo COM con el nombre dado como orden. Si lo encuentra, lo carga en memoria y lo ejecuta. Al acabar el proceso, el sistema nos vuelve a enseñar su indicador y permanece en espera a otra orden. Cuando una orden no puede ser satisfecha nos lo indica mediante un mensaje de error.

Por ejemplo para obtener el directorio del disco B: estando en A: basta escribir DIR B: Con TYPE DATOS. TXT visualizamos en pantalla el contenido del fichero DATOS. TXT residente en el disco dado por defecto. Si tal fichero estaba en B: y el disco activo era A: la orden debería ser TYPE B: DATOS. TXT.

Existen otros comandos del sistema que están en disco, y sólo se cargan en memoria cuando son necesarios: PIP permite hacer una copia de un fichero en el mismo o en otro disco, STAT permite ver el tamaño de cada fichero y el espacio libre de un disco. FORMAT sirve para formatear un disco nuevo para utilizarlo con CPM/80. COPY hace una copia de un disco en otro.

Cada fichero lleva asociado dos atributos que pueden ser (DIR o SYS) y (R/W o R/O), respectivamente para ficheros que aparecen en el directorio o que están ocultos, y para ficheros sin protección contra escritura o con protección. Para ver y cambiar estos atributos se usa también STAT.

Una característica notable es la inclusión de los caracteres libres o comodín * y ? El carácter * concuerda con cualquier sucesión de caracteres, el ? con cualquier carácter. Dan potencia a los comandos del sistema. Por ejemplo para ver un directorio sólo con los ficheros Basic basta escribir DIR *. BAS, para ver todos los ficheros ejecutables con nombre de 3 caracteres DIR ???. COM, y para borrar todos los ficheros cuyo nombre empiece por CT y tengan extensión de 3 letras con X en el centro ERA CT*. ?X?. (Un fichero puede ser borrado, sólo si su atributo de protección es R'W).

Además de las utilidades del sistema, existe una gran biblioteca de programas: lenguajes de programación (MBASIC, BASIC80, CBASIC, PASCAL MT+, FORTRAN 80, COBOL 80, TURBO PASCAL, FORTH, ECO C, ADA, PROLOG, MuLISP), procesadores de texto, hojas de cálculo, bases de datos, programas de comunicaciones,... y programas hechos a medida como facturaciones o contabilidades.

CP/M 80 es simple de usar y aprender, y bastante potente dentro de las limitaciones impuestas por el hardware en que está basado. Posee una gran biblioteca de programas de todo tipo debido a su gran difusión (hasta hace 2 años era el sistema más usado), capaz de satisfacer la mayoría de las necesidades.

Al surgir los microordenadores de 16 bits (con el IBM PC en cabeza), basados en los microprocesadores INTEL 8088, 8086 (y posteriormente 80286) se desarrollan nuevos sistemas operativos que inicialmente están basados en CP/M 80, y después evolucionan para aprovechar las posibilidades que brinda un hardware más potente y avanzado. Son CP/M 86 y MS-DOS (PC DOS).

Digital Research, creadora de CP/M 80, construye una versión para 16 bits llamada CP/M 86, que de cara al usuario es una réplica casi exacta de lo que era la versión anterior. Posteriormente evoluciona hacia la multiárea en forma de Concurrent CP/M y después PC-DOS Concurrente, que más adelante comentaremos.

MS-DOS... el más extendido

Microsoft llega a un acuerdo con IBM para que su sistema operativo MS-DOS sea vendido junto con el microordenador de la multinacional, bajo el nombre PC-DOS. La primera versión de MS-DOS recuerda mucho a CP/M 80, pues el funcionamiento es análogo. Sin embargo, se nota la potencia de la máquina en cuanto a velocidad y mayor direccionamiento de memoria: posibilidad de construir programas muy extensos. Gracias a la amplia difusión del IBM PC, MS-DOS se ha convertido en el sistema operativo más utilizado.

A partir de la versión 2.00 Microsoft, empieza a introducir en MS-DOS (PC-DOS) características provenientes de UNIX, sistema multiusuario multiárea para miniordenadores, con el fin de lograr que las últimas versiones de MS-DOS sean compatibles con XENIX, una versión de UNIX de Microsoft. Lo más relevante es: estructura jerarquizada de los ficheros en forma de árbol, redirección de entradas y salidas y filtros (pipes).

La estructura jerarquizada de ficheros permite la existencia de ficheros especiales llamados subdirectorios que son a su vez listas de ficheros, algunos de los cuales puede ser también un subdirectorio, y así sin límite. Lo que tenemos es una estructura arborescente invertida, con la raíz (denominada ⁄⁄) en la parte superior. Los subdirectorios de un directorio son sus «hijos», mientras que él es el **«padre».**

Gracias a esta estructura es posible hacer agrupaciones lógicas entre ficheros que compartan unas características, por ejemplo conviene tener juntos todas las utilidades del sistema, todos los programas de un lenguaje de programación, documentos escritos con un procesador de textos,... También cuando varias personas utilizan el mismo disco se pueden separar los ficheros que utilizan ambas. Es particularmente útil al utilizar un disco duro en el que pueden caber cientos de ficheros.

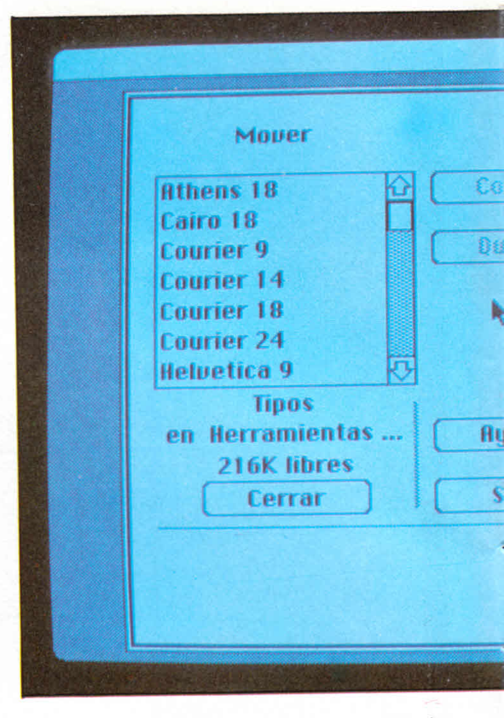
Existen varios comandos para manipular directorios. MKDIR o MD construye un nuevo subdirectorio, hijo del directorio en el que estamos situados. CHDIR o CD nos informa del subdirectorio en que nos encontramos y también nos permite trasladarnos a otro diferente, especificando la cadena de directorios (separados por el carácter ⁄⁄) que hay que recorrer desde la raíz (o desde el mismo) para llegar el directorio deseado, técnicamente el **«pathname».** RMDIR o RD elimina un subdirectorio siempre que dentro de él no exista ningún fichero.

Cuando un programa necesita datos y cuando los imprime lo suele hacer en la consola. MS-DOS permite redireccionar las entradas y salidas, o sea hacer que un programa obtenga su entrada de otro sitio que no sea el estándar, y que imprima sus resultados en otro dispositivo de salida que no sea el estándar. Generalmente se suelen usar ficheros, aunque pueden ser impresoras u otros dispositivos. Se usa el carácter « < » para la entrada y « + » para la salida. Como ejemplo TYPE DATOS.TXT + PRN y TYPE DATOS.TXT + FICH imprimen respectivamente el fichero DATOS.TXT en la impresora, y en un fichero de disco de nombre FICH.

Si tenemos un programa ejecutable de nombre PROG.COM, los comandos PROG < FICH.ENT, PROG + FICH.SAL, y PROG < FICH ENT + FICH.SAL harán que PROG tome la entrada del fichero FICH + ENT y lleve la salida a la pantalla en el primer caso, tome la entrada de la consola y lleve la salida a FICH.SAL en el segundo, y tome la entrada de FICH.ENT y lleve la salida a FICH.SAL en el tercero.

Los filtros son programas que aceptan una entrada, realizan alguna operación con ella y producen una salida. Se pueden encadenar (unidos con el carácter **«I»**) varios filtros («pipeline») de modo que la salida de cada uno alimente la entrada del siguiente. En MS-DOS hay 3 filtros. MORE escribe la entrada en pantalla haciendo pausas entre cada pantalla. FIND escribe en la salida las líneas de la entrada que contenían un cierto fragmento de texto. SORT escribe la salida ordenada. El comando DIR **I** FIND **«.BAT»** I SORT + FICH crea un fichero de nombre FICH conteniendo una lista ordenada de todos los ficheros del directorio cuya extensión sea BAT.

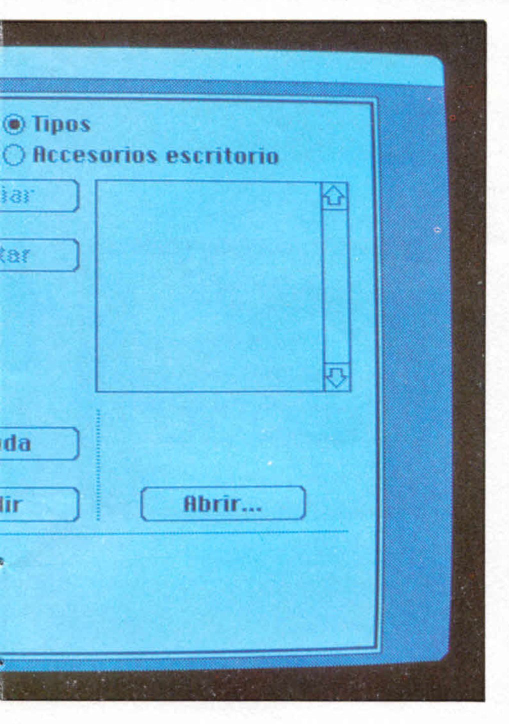
Gracias a la potencia del software se han implementado sistemas multiusuario y multiárea. Un sistema multiusuario es aquel que da servicio a varios usuarios simultáneamente compartiendo sus recursos entre aquéllos. Si es multiárea cada usuario (puede ser multiárea monousuario) puede realizar varios trabajos a la vez. La simultaneidad sólo se consigue con varios procesadores, cuando sólo hay uno se simula repartiendo el tiempo de procesador entre las tareas o usuarios asignándoles a cada una prioridad. El paso de una tarea a otra se suele realizar tan rápidamente que se consigue la sensación de paralelismo.



CCP/M (Concurrent CP/M) y su última versión PC-DOS Concurrent (versión del anterior) ejecutan hasta 4 tareas a la vez, cada una de ellas incluida dentro de lo que se conoce como «consola virtual». Pasar de una tarea a otra equivale a cambiar de consola virtual, apareciendo en pantalla la ejecución del programa de tal consola. Mientras éste se ejecuta (en foreground), los demás procesos siguen su curso (en background). CCP/M es compatible con CP/M 86 y con la mayoría de los programas escritos bajo PC-DOS. Permite como todo sistema multiárea, proteger ficheros para ser compartidos entre usuarios o procesos del mismo usuario, establecer prioridades y crear una ventana en pantalla (para cada consola virtual), que presenta información sobre el proceso asociado, cuando se activa la consola virtual del mismo.

Un lenguaje para un sistema... ¡y viceversa!

Unix es un sistema operativo multiusuario multiárea de miniordenadores, (existen varias versiones para IBM PC), incorpora una estructura jerarquizada de ficheros en forma de árbol invertido, introduce reconducción de entradas y salidas y la existencia de pipelines, poseyendo múltiples filtros, un sofisticado sistema de compartición/protección de ficheros dando los derechos de acceso a un fichero para el mismo usuario, usuarios de un grupo, y



para el resto de los usuarios. Posee más de 200 utilidades, entre las que se suelen incluir un compilador de C y de FORTRAN 77, que son una gran fuente de ayuda para el programador.

Unix está muy ligado al lenguaje C, ya que de las 13000 líneas de código, unas 800 se escribieron en ensamblador para satisfacer las necesidades de más bajo nivel, el resto se escribió en C, al igual que la mayoría de las utilidades.

Unix demuestra que hasta un sistema operativo puede escribirse con un lenguaje de relativo alto nivel como C. Gracias a esto es muy fácil transportar un sistema Unix a una máquina nueva.

Cuando se accede al sistema entra en funcionamiento un programa llamado Shell, que es el encargado de reconocer las órdenes emitidas, ejecutando los comandos.

También permite al usuario escribir programas en un lenguaje estructurado propio de gran potencia y flexibilidad.

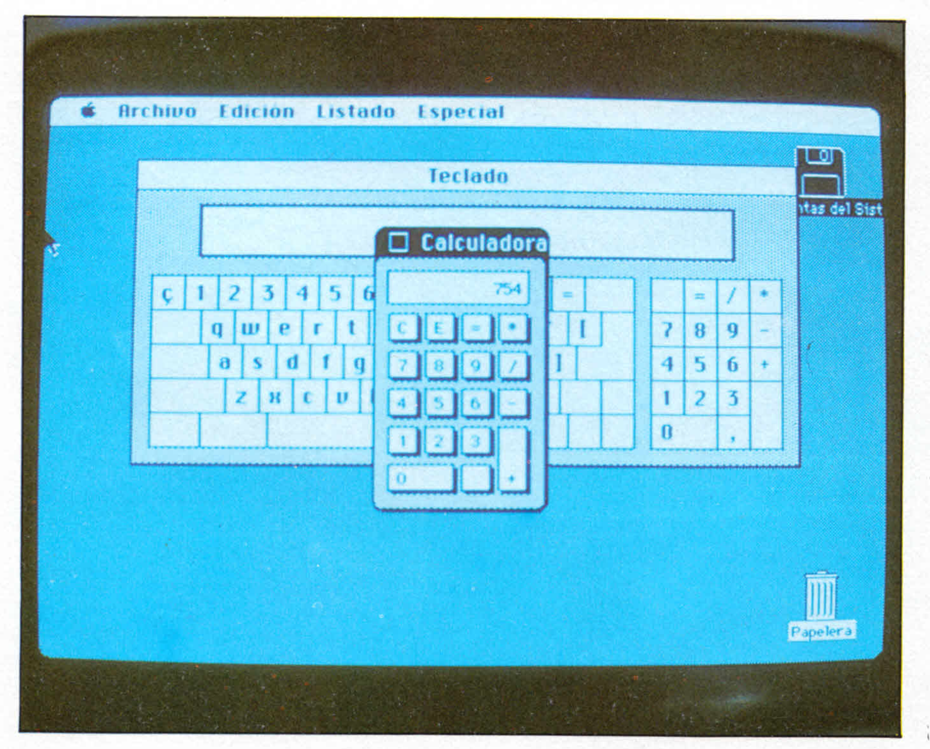
La multiárea hace que los programas corran más lentamente que si se ejecutaran individualmente, pero proporcionan otras ventajas. Aprovecha los tiempos muertos en espera de datos o en entradas y salidas a dispositivos, y siempre está a disposición del usuario.

El usuario nunca tiene que esperar a que finalice un proceso para poder realizar otra cosa, la máquina está disponible en cualquier momento. La concurrencia se adapta mejor a la forma de trabajar del hombre, el ejemplo típico es el programador que simultáneamente compila un programa, imprime el fichero fuente y hace modificaciones en el mismo. Cada tarea irá lenta que ejecutada por separado, pero la ventaja de proceder así es evidente.

El sistema operativo del MacIntosh se diferencia de los sistemas anteriores en el interface con el usuario, que no se relaciona con procedimientos sino con objetos, representados gráficamente en la pantalla del ordenador mediante «iconos», que son manejados con un ratón, y cuya imagen guarda relación con la operación que realiza. Así, en vez de tener que recordar una larga serie de comandos con una sintaxis a veces compleja, en el MacIntosh sólo se debe mirar la pantalla, elegir el icono que represente la acción a realizar y seleccionarlo con el ratón. El usuario se concentra en la tarea que está haciendo y no en los pasos que debe seguir para realizarla. Para borrar un fichero, basta señalarlo con la flecha (moviendo el ratón), pulsar el botón del ratón, y sin soltarlo trasladarlo al icono que representa la papelera.

Al encender el equipo aparece en pantalla una boca de disco indicando que se debe introducir alguno. Después aparece dibujado un disco y un menú de opciones. Mediante el ratón podemos poner la flecha sobre una opción y aparece una ventana con una lista de acciones posibles. Para seleccionar un disco se lleva la flecha encima de su icono y se pulsa el botón del ratón, entonces se ilumina el icono del disco indicando que la acción ha sido comprendida y realizada. Moviendo la flecha al menú de opciones y seleccionando FILE apretando el botón sin soltarlo, aparecerá una lista de acciones relacionadas con el manejo de ficheros. Para elegir OPEN ponemos encima la flecha y soltamos el botón del ratón, así aparece un directorio con los ficheros que contiene el disco, cada uno representado con un icono alusivo y su nombre debajo. Si seleccionamos un fichero de texto, el MacIntosh piensa que lo queremos editar y carga su procesador de textos Mac Write y el fichero seleccionado. Apuntando a un fichero ejecutable se inicia la ejecución del mismo.

El sistema operativo incluye todo el material necesario para desarrollar el software que permita utilizar todo el entorno gráfico, desde el manejo de ventanas hasta un surtido juego de tipos (y tamaños) de letra.



Tras las huellas de Mac

Digital Research ha creado GEM un entorno operativo que rodea a MS-DOS y sirve de interface con el usuario al estilo MacIntosh, eliminando la necesidad de teclear los comandos habituales del sistema, mediante el uso de ventanas e iconos. Tan similar es este entorno al del MacIntosh, que GEM ha sido rediseñado para evitar problemas legales con Apple Computer dueña del sistema del MacIntosh.

También existen versiones de GEM para los nuevos ordenadores ATARI ST y AMIGA, que nuevamente actúan como recubrimientos del verdadero sistema operativo. Hay que observar que si un programa no ha sido diseñado para trabajar con GEM, aunque pueda ser ejecutado desde éste, se comportará como siempre.

El uso de iconos, ventanas y ratones se popularizó hace dos años gracias a la difusión del MacIntosh, pero tienen más de diez años de vida, y se remontan a los trabajos realizados por la compañía RANK XEROX (a su vez inspirados en estudios teóricos) que los implementó por primera vez en su ordenador STAR, muy avanzado para su época. De aquí surge también el lenguaje de programación orientada al objeto SMALLTALK, que utiliza profusamente ventanas, iconos y ratón.

AMSTRAD CPC - 464

AMSTRAD



ORDENADORI

SERIE CPC

UNIDAD CENTRAL. MEMORIAS
• Microprocesador Z80A - 64K RAM ampliables - 32K ROM ampliables
• TECLADO • Teclado profesional con 74 teclas en 3 bloques - Hasta 32 teclas programables - Teclado redefinible
• PANTALLA • Monitor RGB verde (12") o color (14")

	Normal	Alta Res.	Multicolor
Col × líneas	40 × 25	80 × 25	20 × 25
Colores	4 de 27	2 de 27	16 de 27
Puntos	320 × 200	640 × 200	160 × 2

– Se pueden definir hasta 8 ventanas de texto y 1 de gráficos **• SONIDO**
• 3 canales de 8 octavas moduladas independientemente - Altavoz interno regulable - Salida estéreo **• BASIC**
• Locomotive BASIC ampliado en ROM - Incluye los comandos AFTER y EVERY para control de interrupciones

AMSTRAD CPC 464

CASSETTE • Cassette incorporada con velocidad de grabación (1 ó 2 Kbaudios) controlada desde Basic **• CONECTORES**
• Bus PCB multiuso, Unidad de Disco exterior, paralelo Centronics, salida estéreo, joystick, lápiz óptico, etc.
• SUMINISTRO • Ordenador con monitor verde o color - 8 cassettes con programas - Libro "Guía de Referencia BASIC para el programador" - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR	59.900 Pts. (monitor verde)
	90.900 Pts. (monitor color)

AMSTRAD CPC 6128

UNIDAD DE DISCO • Unidad incorporada para disco de 3" con 180K por cara **• SISTEMAS OPERATIVOS**
• AMSDOS, CP/M 2.2, CP/M Plus (3.0)
• CONECTORES • Bus PCB multiuso, paralelo Centronics, cassette exterior, 2.ª Unidad de Disco, salida estéreo, joysticks, lápiz óptico, etc.
• SUMINISTRO • Ordenador con monitor verde o color - Disco con CP/M 2.2 y lenguaje DR. LOGO - Disco con CP/M Plus y utilidades - Disco con 6 programas de obsequio - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR	84.900 Pts. (monitor verde)
	119.900 Pts. (monitor color)

PCW - 8256 AMSTRAD CPC - 6128

ES AMSTRAD ¡¡Increíble!!

AMSTRAD PCW 8256

UNIDAD CENTRAL. MEMORIAS • Microprocesador Z80A - 256K RAM de las que 112K se utilizan como disco RAM • **TECLADO** • Teclado profesional en castellano (ñ, acento...) de 82 teclas • **PANTALLA** • Monitor verde de alta resolución - 90 columnas × 32 líneas de texto • **UNIDAD DE DISCO** • Disco de 3" y 173K por cara - Opcionalmente, 2.ª Unidad de Disco de 1 Mbyte integrable • **SISTEMA OPERATIVO** • CP/M Plus de Digital Research • **IMPRESORA** • Alta calidad (NLQ) a 20 c.p.s. - Calidad estándar a 90 c.p.s. - Papel continuo u hojas sueltas - Alineación automática del papel - Caracteres normales, comprimidos, expandidos, control del paso de letra (normal, cursiva, negrita, subíndices, superíndices, subrayado, etc). • **OPCIONES** • Kit de Ampliación a 512K RAM y 2.ª Unidad de Disco - Interface Serie RS 232C y paralelo Centronics • **SUMINISTRO** • Ordenador completo con teclado, pantalla, Unidad de Disco e Impresora - Discos con el procesador de Texto LocoScript, CP/M Plus, Mallard, BASIC, DR. LOGO y diversas utilidades - Manuales en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR 129.900 Pts.



Los más prestigiosos paquetes de **Software Profesional,** en formato AMSTRAD... a **"precios AMSTRAD"**

Existe también la versión **PCW 8512** con **512K RAM** y la 2.ª Unidad de Disco de 1 Mbyte incorporada. **PVP. 169.900 Pts.**

* El **PCW 8256** puede utilizarse como terminal y en comunicaciones.

El I.V.A. no está incluido en los precios.

NOTA: Es muy importante verificar la garantía del aparato ya que sólo **AMSTRAD ESPAÑA** puede garantizarle la ordenada reparación y sobre todo materiales de repuesto oficiales (Monitor, ordenador, cassette o unidades de discos).

AMSTRAD ESPAÑA

Avda. del Mediterráneo, 9. Tels. 433 45 48 - 433 48 76. 28007 MADRID

Delegación Cataluña: Tarragona, 110 - Tel. 325 10 58. 08015 BARCELONA

GNOMOS

¿Conoces a David el gnomo?, sí, pues sus simpáticos amigos han venido a jugar con nosotros con su arbolito de frutas. Se debe recoger todas entre ellos y tú, pero ten cuidado de no recoger la última, porque si no habrás perdido. Demuéstrales que eres más hábil que ellos.



Este programa constituye un juego inteligente, durante el cual debemos enfrentarnos a un duendecillo que controla el ordenador, o bien a un jugador humano.

Se trata de una versión modificada del juego del Nim, en pantalla aparecerán una serie de vegetales comestibles, algunos crecen sobre un árbol, y otros del suelo.

Cada jugador deberá elegir por turno qué tipo de vegetal desea eliminar de entre los cinco posibles (manzanas, peras, hojas, cerezas, setas), así como la cantidad a suprimir.

En cada turno sólo está permitido eliminar un tipo, es decir, podemos suprimir si queremos 4 hojas o 3 manzanas, o 2 peras, pero nunca 2 hojas y 3 manzanas en la misma jugada, por ejemplo.

El jugador que elimine el último de los vegetales que quede en la pantalla, pierde automáticamente la partida.

Ejemplo: Si en pantalla hay 3 cerezas y ningún vegetal más, yo eliminaría 2 cerezas, mi contrincante se vería obligado a eliminar la última que quedara y yo ganaría la partida.

Ejemplo: Si en pantalla quedan 3 manzanas y una seta, yo suprimiría las 3 manzanas, mi contrincante se encontraría con que sólo puede eliminar la seta que resta y yo ganaría la partida.

Instrucciones de manejo

Al principio se presentan dos opciones.

A Un jugador (es decir humano contra máquina).

B Dos jugadores (dos humanos se enfrentan entre sí).

Si escogemos la opción A, debemos decidir contra qué duende de los cinco que se nos presentan nos enfrentaremos. Para ello basta pulsar el dígito correspondiente.

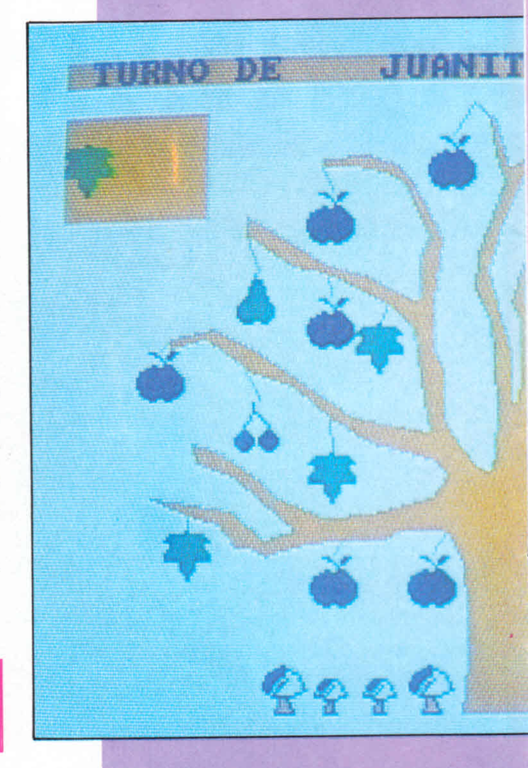
Mientras más alto sea el número del duende escogido mayor será la dificultad.

Una vez impresa la pantalla correspondiente debemos proceder así.

— Seleccionar uno de los vegetales mediante la tecla espacio.

— Teclear una cifra, el ordenador entenderá que corresponde a la cantidad de unidades a eliminar del vegetal seleccionado.

— Pulsar intro para ejecutar la jugada.



Estructura del programa

Rutina 1. Líneas 10-640

Asigna colores a plumas y borde. Define ventanas 1 y 2. Dimensiona





variables. Define caracteres gráficos. Asigna valores a variables de cadena. Asigna valores a variables mediante sentencias data.

Rutina 2. Líneas 650-1080

Presentación del juego. Impresión del menú. Se establece número de jugadores. Nombre de jugadores. Impresión. Se define grado de dificultad.

Rutina 3. Líneas 1090-1290.

El ordenador decide al azar quién comienza. Realización del dibujo de árbol y gnomo. Paso a subrutina n.° 4 para dibujar setas.

Rutina 4. Líneas 1300-1410

Imprime mediante un bucle los vegetales que cuelgan del árbol. Pulsando una tecla paso a rutina n.° 5.

Rutina 5. Líneas 1420-1550

Bucle principal del juego. Imprime en pantalla nombre del jugador en turno. Si le toca turno a humano pasa a subrutina n.° 1. Si es turno de ordenador paso a subrutina n.° 2. Si acaba la partida remite a rutina 2 tras imprimir nombre del ganador.

Subrutina 1. Líneas 1560-1780

Establece la jugada de humano. Imprime en ventana gráfica el vegetal elegido y la cantidad a suprimir del mismo. Si la jugada es factible, pasa a subrutina 3. Vuelta al bucle principal.

Subrutina 2. Líneas 1790-2110

El ordenador establece la jugada a realizar. Una vez elegida la jugada pasa a subrutina 3. Vuelta a bucle principal.

Subrutina 3. Líneas 2120-2340

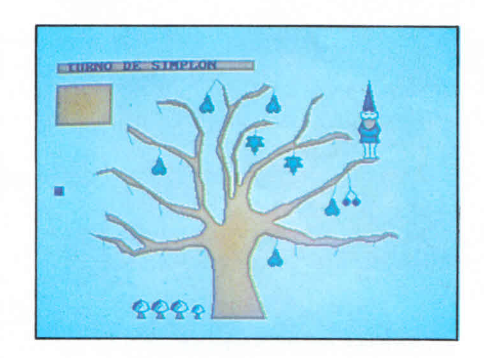
Realiza las jugadas de las subrutinas 1 ó 2. Sigue las instrucciones de las subrutinas citadas. Explosionando y borrando los vegetales escogidos.

Subrutina 4. Líneas 2350-2430

Realiza el dibujo de una seta. Dependiendo del valor de la variable CO, retorna a rutina 3 o A subrutina 1.

Lista de variables principales

PRO: Número de jugadores humanos (1 ó 2).
COM: Grado de complejidad del juego (1 a 5).
JA: Su valor indica a quién corresponde el turno de juego (1 a 2).
ME: Durante el desarrollo de la partida indica qué tipo de vegetal debe suprimirse en la pantalla, tanto si juega humano como máquina.



D: Especifica el número de unidades a explosionar y retirar de pantalla del vegetal especificado por la variable ME.
NU (n): Son 22 variables que indican el tipo de fruta que existe en cada posición del árbol al comenzar la partida.
CU (n): Son 5 variables que indican la cantidad total de cada vegetal que existe en pantalla en ese momento.
Cu (1)... Cantidad de hojas.
Cu (2)... Cantidad de peras.
Cu (3)... Cantidad de manzanas.
Cu (4)... Cantidad de cerezas.
Cu (5)... Cantidad de setas.

Otras variables

ti: Indica la tinta a usar para imprimir los caracteres gráficos.
c: Especifica tamaño de la seta.
CLAVE: Nos informa de si el ordenador encontró ya una jugada correcta, o debe seguirla buscando.

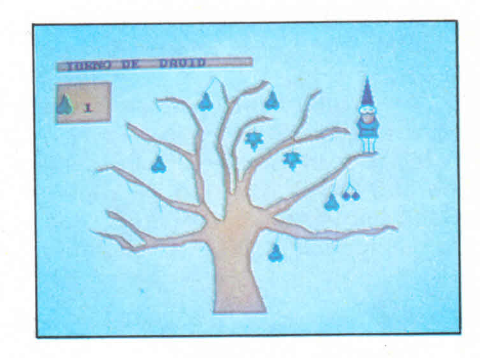
O y H: Se utilizan para establecer un bucle que es el encargado de borrar las setas.

e (n): Similar a cu (n), pero la utiliza la máquina cuando se juega contra ella para calcular la jugada correcta.

JU$ (n): Contienen nombre jugadores humanos.

GNO$ (N): Contienen los nombres de los cinco gnomos.

BI $, p$ (n) y p (n): La primera es



similar A e (n) pero en binario, p$ (n) son los sucesivos dígitos de BI$ (n), p (n)=VAL p$ (n).

SUM: Almacena el valor de e (1)+e (2)+e (3)+e (4)+e (5).

b$ (n): Esta variable almacena el valor de las letras de JU$ (n).

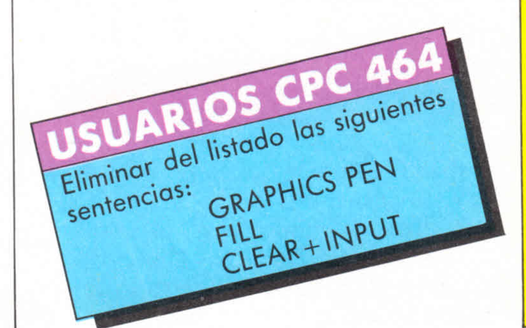
Variables relacionadas con los gráficos

Pe$ Ra$ Man$ Za$ Ce$ Re$ Ho$ Ja$ Ex$ plo$. Contienen los caracteres definidos para imprimir pera, manzana, hoja, cereza y hacer el efecto de explosión respectivamente.

Ar (n) y Bol (n). Almacenan las coordenadas de los 227 puntos que se utilizan para imprimir el árbol.

Se (n), Ta (n), Gno (n) y Mo (n) son similares a las anteriores pero sirven para dibujar la seta y el gnomo respectivamente.

Si (n) Tio (n). Especifican la fila y columna donde se deben imprimir los 22 vegetales que cuelgan del árbol.




```
10 SYMBOL AFTER 200
20 INK 0,15: INK 1,3: INK 2,23: INK
 3,9:PAPER 2: CLS: BORDER 23
30 WINDOW #2,1,6,3,6
40 WINDOW #1,1,21,1,1
50 DIM ar(227): DIM bol(227)
60 DIM se(23): DIM ta(23)
70 DIM gno(54): DIM mo(54)
80 DIM si(22): DIM tio(22)
90 DIM hon(4): DIM nu(22)
100 DIM cu(5):DIM ju$(2):DIM gno$(5
)
110 DIM bi$(5): DIM p$(5): DIM p(5)
: DIM e(5):DIM b$(11)
120 SYMBOL 200,1,1,3,3 ,7,7,7,15
130 SYMBOL 201,0,0,192,192,224,224,
224,240
140 SYMBOL 202,31,63,127,127,127,12
7,63,30
150 SYMBOL 203,240,248,252,252,252,
252,248,112
160 SYMBOL 204,14,3,0,0,6,31,127,12
7
170 SYMBOL 205,28,56,176,64,112,248
,252,254
180 SYMBOL 206,255,255,255,255,127,
127,63,14
190 SYMBOL 207,255,255,255,255,255,
254,252,112
200 SYMBOL 208,2,2,2,3,4,4,8,16
210 SYMBOL 209,0,0,0,0,128,64,32,16
220 SYMBOL 210,16,56,124,254,254,25
4,124,56
230 SYMBOL 211,1,63,127,255,63,63,1
27,255
240 SYMBOL 212,128,252,254,255,252,
252,254,255
250 SYMBOL 213,255,255,7,7,7,3,1,1
260 SYMBOL 214,255,255,224,224,224,
192,128,128
270 SYMBOL 215,2,130,64,12,156,73,1
,28
280 SYMBOL 216,8,16,33,0,132,144,15
2,88
290 SYMBOL 217,12,26,0,32,2,128,8,0
300 SYMBOL 218,16,0,228,98,1,64,64,
0
310 pe$=CHR$(200)+CHR$(201): ra$=CH
R$(202)+CHR$(203)
320 man$=CHR$(204)+CHR$(205): za$=C
HR$(206)+CHR$(207)
330 ce$=CHR$(208)+CHR$(209):re$=CHR
$(210)+CHR$(210)
340  ho$=CHR$ (211)+CHR$(212): ja$=
CHR$(213)+CHR$(214)
350 ex$=CHR$(215)+CHR$(216)
360 plo$=CHR$(217)+CHR$(218)
370 gno$(1)="SIMPLON":gno$(2)="TONT
ILLO":gno$(3)="LISTORRO":gno$(4)="M
EMORION":gno$(5)="SABIONDO"
380 FOR n= 1 TO 227
390 READ ar(n),bol(n)
400 NEXT n
410 DATA 262,30,262,60,260,90,250,1
10,241,96,250,110,220,110,210,105,1
80,105,177,96,180,105,160,102,150,1
00,130,95,120,100,110,100,100,115,8
0,120,81,112,80,120,56,130,110,122,
130,105,150,110,128,128,110,145,87,
152,85,165,100,160
420 DATA 130,140,140,130,170,118,20
0,120,240,130,250,145,240,155
430 DATA 200,165,175,180,177,160,17
5,180,150,200,120,210,129,192,120,2
10,95,230,70,225,70,230,65,224,70,2
30,100,236,130,220,160
440 DATA 200,190,180,220,174,240,17
0,236,190,230,235,215,250,209,240,2
15,250,190,262,175,270,177,256,175,
270,150,275,120,295,129,272,120,295
,120,305,130,300,175,275,200,268,23
5,250,235,280,240,290,225,325
450 DATA 210,330,190,335,170,335,17
7,320,170,335, 170,338,200,340,220,
335,230,325,250,290,245,250,245,220
,250,210,250,180,255,170,270,150,28
0,145,285,180,282,220,280,240,270,2
60,272,280,290,325,285,340,280,360,
270,370,257,352,270,370,285,360,295
,330
460 DATA 320,350,340,355,350,365,36
5,360,369,352,365,360,340,345,315,3
40
470 DATA 300,320,290,300,280,270,29
5,220,295,180,300,200,300,240,295,2
50,300,270,310,290,330,305,350,310,
375,305,350,300,337,299,337,288,337
,299,320,290,305,260,310,200,315,22
0,330,240,350,250,370,265,390,275,4
00,280,415,300,420,320,430,340
480 DATA 450,350,450,346,430,330,42
5,300,400,270,430,280,440,280,460,2
90,475,293,490,290,500,280,470,282,
465,272,470,282,450,275,430,270,400
,264,401,256,400,264,390,260,370,25
0,340,230,330,215,325,190,330,170,3
50,165,380,180,388,180,420,195
490 DATA 440,205,460,218,480,230,49
0,245,510,250,530,252,540,254,545,2
50,530,245,510,245,490,230,492,208,
490,230,470,210,465,192,470,210,450
,200,420,185,400,170,390,160,365,15
0,390,140,410,137,430,125
500 DATA 455,128,480,125,510,128,54
0,130,550,135,560,135,570,130,570,1
25,555,128,540,120,530,118,529,112,
530,118,500,120,470,110,465,96,470,
110,450,113,430,113,401,115,401,96,
401,115,390,120,370,125,369,112,370
,125,350,130,340,125,325,90,325,70,
 350,0
510 FOR x=1 TO 22
520 READ si(x):READ tio(x)
530 NEXT x
540 DATA 33,19,29,20,25,20,27,19,23
,19,11,20,15,20,5,19,11,16,8,14,4,1
2,13,11,11,10,8,9,11,6,16,4,23,4,21
,8,29,9,25,10,29,14,31,13
550 FOR x=1 TO 23
560 READ se(x),ta(x)
570 NEXT x
580 DATA 3,10,2,10,10,0,-10,0,-5,0,
-5,2,-10,3,1,5,4,10,5,5,5,3,10,2,10
,-2,5,-3,5,-5,4,-10,1,-5,-10,-3,-5,
-2,-5,0,2,-10,3,-10,-20,0
590 FOR n= 1 TO 54
600 READ gno(n), mo(n)
610 NEXT n
620 DATA -12,-48,12,4,-12,-4,-2,0,0
,-6,8,-4,6,2,-6,-2,-8,4,2,-6,4,-10,
8,-4,-8,4,-4,10,-6,-14,4,-8,14,-2,-
14,2,-2,-12,4,0,12,0,-12,0,4,-10,0,
-8,-2,-6,10,0,0,24
630 DATA 0,-24,10,0,-2,6,0,8,4,10,-
12,0,12,0,4,0,-2,12,-14,-2,14,2,4,8
,-6,14,-4,-10,-8,-4,8,4,4,10,2,6,-8
,-4,-6,2,6,-2,8,4,0,6,-2,0,-12,4,12
,-4,-12,48
640 hon (1)=136: hon(2)=168: hon(3)
=200:hon (4)=232
650 ju$(1)="": ju$(2)=""
660 FOR n=100 TO 500 STEP 100
670 MOVE n,200
680 FOR x=1 TO 54
690 DRAWR gno(x),mo(x)
700 NEXT x
710 MOVER 0,-10:FILL 1:MOVER 0,-50:
FILL 0: MOVER 0,-20:FILL 3: MOVER 0
,-10: FILL 3
720 PLOT n-5,150:DRAWR 2,0: PLOT n+
4,150: DRAWR 3,0
730 NEXT n
740 LOCATE 10,2:PRINT "MENU"
750 LOCATE 10,4:PRINT "A....UN JUGA
DOR"
760 LOCATE 10,5:PRINT "B....DOS JUG
ADORES": CLEAR INPUT
770 a$=INKEY$:a$=UPPER$ (a$)
780 IF a$<>"A" AND  a$<>"B" THEN GO
TO 770
790 IF a$="A" THEN pro=1: LOCATE 5,
4:PRINT STRING$(2,143)
800 IF a$="B" THEN pro=2: LOCATE 5,
5:PRINT STRING$(2,143)
810 FOR n=1 TO pro
820 FOR z=1 TO 11: b$(z)=" ":NEXT z
830  x=1
840 LOCATE 5,7:PRINT "NOMBRE JUGADO
R";n
```

```
850 LOCATE 5,9:PRINT STRING$(9,127)
860 a$=INKEY$:IF a$="" THEN GOTO 86
0
870 IF INKEY( 6)=0 THEN GOTO 940
880 IF INKEY( 79)=0 AND x>1 THEN x=
x-1:b$(x+1)=" ":LOCATE x+4,9:PRINT
CHR$(127):GOTO 860
890 IF INKEY(79)=0 THEN CLEAR INPUT
:GOTO 860
900 a$=UPPER$(a$): FOR Z= 1 TO 200:
 NEXT Z
910 LOCATE x+4,9:PRINT a$:x=x+1:b$(
x)=a$
920 IF x=10 THEN GOTO 940
930 GOTO 860
940 FOR x=1 TO 10: ju$(n)=ju$(n)+b$
(x):NEXT x
950 NEXT n
960  IF pro=1 THEN GOTO 970 ELSE GO
TO 1080
970 LOCATE 1,2:PRINT JU$(1);" POR
FAVOR   ELIGE ": LOCATE 2,4:PRINT "
EL NUMERO DE TU CONTRINCANTE"

980 LOCATE 1,5: PRINT STRING$(38,12
8)
990 FOR n= 7 TO 9 STEP 2: LOCATE 1,
n:PRINT STRING$(38,127): NEXT n
1000 LOCATE 2,22: PRINT "1.";gno$(1
)
1010 LOCATE 9,24: PRINT "2.";gno$(2
)
1020 LOCATE 15,22: PRINT "3.";gno$(
3)
1030 LOCATE 23,24:PRINT "4.";gno$(4
)
1040 LOCATE 30,22:PRINT "5.";gno$(5
)
1050 A$=INKEY$
1060 IF A$<>"1" AND A$<>"2" AND A$<
>"3" AND A$<>"4" AND A$<>"5" THEN G
OTO 1050
1070 COM=VAL (A$)
1080 CLS
1090 GRAPHICS PEN 0: MOVE 258,0: o=
9
1100 ja=INT(RND*2)+1
1110 FOR n=1 TO 227
1120 DRAW ar(n),bol(n)
1130 NEXT n
1140 MOVE 300,0: FILL 0
1150 GRAPHICS PEN 1
1160 FOR n= 1 TO 4
1170 MOVE hon(n),1
1180  c= (4+INT(RND*2))/10
1190 GOSUB 2350
1200 NEXT n
1210 MOVE 526,370
1220 FOR n=1 TO 54
1230 DRAWR  gno(n),mo(n)
1240 NEXT n
1250 MOVER 0,-10: FILL 1
1260 MOVER 0,-50: FILL 0
1270 MOVER 0,-20: FILL 3
1280 MOVER 0,-10: FILL 3
1290 PLOT 521,320: DRAWR 2,0: PLOT
530,320: DRAWR 3,0
1300 CLEAR INPUT: PRINT #1, "PULSA
UNA TECLA":
1310 FOR x=1 TO 4: cu(x)=0:NEXT x:
cu(5)=4
1320 FOR x=1 TO 22
1330 SOUND 1,25,1,5
1340 RANDOMIZE TIME: nu (x)= INT (R
ND*4)+1
1350 ON nu(x) GOTO 1360,1370,1380,1
390
1360 pri$=ho$:seg$=ja$:cu(1)=cu(1)+
1: ti=3: GOTO 1400
1370 pri$=pe$:seg$=ra$:cu(2)=cu(2)+
1: ti=3: GOTO 1400
1380 pri$=man$:seg$=za$:cu(3)=cu(3)
+1: ti=1: GOTO 1400
1390 pri$=ce$:seg$=re$:cu(4)=cu(4)+
1: ti=1: GOTO 1400
1400 PEN  ti: LOCATE si(x),tio(x):
PRINT pri$: LOCATE si(x),( tio(x)+1
): PRINT seg$
1410 NEXT x:IF INKEY$<>"" THEN GOTO
 1420 ELSE GOTO 1310
```

```
1420 IF cu(1)+cu(2)+cu(3)+cu(4)+cu(
5)<2 THEN GOTO 1480
1430 IF ja=1 AND pro=2 THEN PRINT #
1, " TURNO DE ";ju$(2)
1440 IF ja=1  AND pro=1 THEN PRINT
#1," TURNO DE ";gno$(com):CLS #2
1450 IF ja=2 THEN PRINT #1, " TURNO
 DE ";ju$(1)
1460 FOR x=1 TO 400: NEXT x
1470 IF ja=2 OR pro=2 THEN GOTO 156
0 ELSE GOTO 1790
1480 FOR x=1 TO 1500: NEXT x: CLS
1490 IF CU(1)+CU(2)+CU(3)+CU(4)+CU(
5)=0 THEN JA=JA+1:IF JA=3 THEN JA=1
1500 IF PRO=2 THEN PRINT "GANADOR..
..........";ju$(ja)
1510 IF pro=1 AND ja=1 THEN PRINT "
GANADOR.." JU$(1):GOTO 1540
1520 IF PRO=1 AND ja=2 THEN PRINT "
GANADOR.." gno$(com)
1530 IF com>1 AND pro=1 THEN LOCATE
 1,4:PRINT ju$(1);" TU PROXIMO CONT
RINCANTE ":LOCATE 1,5: PRINT " DEBE
 SER ";gno$(com-1)
1540 LOCATE 1,10: PRINT "PULSE S,SI
 DESEA JUGAR OTRA PARTIDA"
1550 IF INKEY(60)=0 THEN CLS: GOTO
650 ELSE GOTO 1550
1560 me=1:D=0: IN=0:CLEAR INPUT: GO
TO 1600
1570 IF INKEY(47)=0 THEN d=0: me=me
+1:GOTO 1600
1580 IF INKEY(6)=0 AND d>0 THEN GOT
O 1740
1590 in$=INKEY$: IF in$="0" OR in$=
"1" OR in$="2" OR in$="3" OR in$="4
" OR in$="5" OR in$="6" OR in$="7"
OR in$="8" OR in$="9" THEN GOTO 170
0 ELSE GOTO 1570
1600 IF me=6 THEN me=1
1610 CLS#2: ON me GOTO 1620,1640,16
30,1650,1690
1620 pri$=ho$:seg$=ja$: ti=3:GOTO 1
660
1630  pri$=man$: seg$=za$: ti=1: GO
TO 1660
1640 pri$=pe$:seg$=ra$: ti=3:GOTO 1
660
1650 pri$=ce$:seg$=re$: ti=1: GOTO
1660
1660  LOCATE #2,1,2:PEN #2, ti: PRI
NT #2, pri$
1670 LOCATE #2,1,3: PEN #2,  ti: PR
INT #2, seg$
1680 FOR x= 1 TO 100: NEXT x: GOTO
1570
1690 MOVE 9,321: co=1:c=0.5: GOSUB
2350: co=0: GOTO 1570
1700 in=VAL (in$): d=(d*10)+in
1710 IF d> 99 THEN d=0: GOTO 1560
1720 LOCATE #2,3,3:PEN #2,1: PRINT
#2,d
1730 FOR x=1 TO 100: NEXT x: GOTO 1
570
1740 IF d>cu(me) THEN GOTO 1560
1750 cu(me)=cu(me)-d
1760 GOSUB 2120
1770 IF ja=1 THEN ja=2:GOTO 1420
1780 IF ja=2 THEN ja=1:GOTO 1420
1790 clave =0
1800 FOR x= 1 TO 5
1810  E(X)=CU(X)
1820 NEXT x
1830 IF cu(1)+cu(2)+cu(3)+cu(4)+cu(
5)>(com*5)+1 THEN :FOR w=1 TO 700:N
EXT w:  GOTO 1980
1840 IF e(1)<2 AND e(2)<2 AND e(3)<
2 AND e(4)<2 AND e(5)<2 THEN GOTO 1
850 ELSE GOTO 1870
1850 sum= e(1)+e(2)+e(3)+e(4)+e(5)+
1
1860 ON sum GOTO 2030,  1980,2030,1
980,2030,1980
1870 FOR x=1 TO 5
1880 bi$(x)= BIN$(e(x),8)
1890 NEXT x
1900 FOR n=1 TO 8
1910 FOR x=1 TO 5
1920 p$(x)= MID$(bi$(x),n,1)
1930 p(x)= VAL (p$(x))
```




```
1940 NEXT x
1950 sum=p(1)+p(2)+p(3)+p(4)+p(5)
1960 IF sum=1 OR sum=3 OR sum=5 THE
N GOTO 2030
1970 NEXT n
1980 IF clave=1 THEN GOTO 2090
1990 me= INT (RND*5)+1
2000 IF cu(me)=0 THEN GOTO 1990
2010 d= INT (RND*cu(me))+1
2020 GOTO 2100
2030 clave =1
2040 FOR x=1 TO 5: e(x)=cu(x):NEXT
x
2050 me=INT (RND*5)+1
2060 IF cu(me)=0 THEN GOTO 2050
2070 d= INT (RND*cu(me))+1
2080 e(me)=e(me)-d : GOTO 1840
2090 GOTO 2100
2100  cu(me)=cu(me)-d
2110 GOSUB 2120: ja=2: GOTO 1420
2120 IF me=5 THEN GOTO 2280
2130  FOR X=1 TO 22
2140 IF D=0 THEN RETURN
2150 IF nu(x)=me THEN  GOTO 2170
2160 NEXT x: GOTO 2270
2170 d=d-1: nu(x)=0
2180 IF me>3 THEN ti=1
2190 IF me<3 THEN ti=3
2200 PEN ti
2210 LOCATE si(x),tio(x):PRINT ex$
2220 LOCATE si(x),tio(x)+1:PRINT pl
o$
2230 ENV 1,1,14,1,7,-2,4:SOUND 1,0,
-1,0,1,0,1:  FOR n=1 TO 200: NEXT n
2240 LOCATE si(x),tio(x):PRINT "  "
2250 LOCATE si(x), tio (x)+1: PRINT
 "  "
2260 GOTO 2160
2270 RETURN
2280 FOR H=o TO 15-(cu(me)*2) STEP
2
2290  LOCATE H,24: PRINT ex$
2300 LOCATE H,25: PRINT plo$
2310 ENV 1,1,14,1,7,-2,4:SOUND 1,0,
-1,0,1,0,1:  FOR n=1 TO 200: NEXT n
2320 LOCATE h,24: PRINT "  "
2330 LOCATE h,25: PRINT "  "
2340 NEXT H:o=17-(cu(me)*2): RETURN
2350 FOR X= 1 TO 23
2360 DRAWR se(x)*c,ta(x)*c
2370 NEXT x
2380 IF co=1 THEN RETURN
2390 MOVE  hon(n)+(5*c),1+(40*c): D
RAWR 5*c,5*c:DRAWR 5*c,-5*c: DRAWR
-5*c,-5*c: DRAWR -5*c,5*c
2400  MOVE  hon(n)-(5*c),1+(30*c):
DRAWR 5*c,5*c:DRAWR 5*c,-5*c: DRAWR
 -5*c,-5*c: DRAWR -5*c,5*c
2410 MOVE hon(n),1+(40*c): FILL 1
2420  MOVE hon(n)+3,3: FILL 0
2430 RETURN
```



Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicitanoslo.

Ofites

Presenta: el universo del software, y

DELTA +

La más moderna base de datos DELTA, superándose a sí misma, "DELTA +", desarrollada para CP/M por COMPSOFT con todo en español.

Diseña sus propios ficheros; desde un simple fichero de nombres y direcciones hasta su propio sistema contable. El formato standar DIF permite intercambiar datos en DELTA, desde las hojas de cálculo CRACKER II, etc... y viceversa. Intercambio de datos con la mayoría de los tratamientos de texto como NEWWORD para MAILING.

Incluye un sencillo y funcional sistema de impresión de etiquetas con: hasta 5 columnas de etiquetas, 65 caracteres por etiquetas, 20 líneas con 3 campos cada una.

- PROGRAMABLE Y RELACIONAL.
- FICHEROS INDEXADOS.
- HASTA 90 CAMPOS ó 2.000 CARACTERES.
- MULTIPLES SISTEMAS DE BUSQUEDA, 8 CLAVES.
- FICHEROS DE HASTA 8 Mb.
- 8 GRUPOS DE TRANSACCION POR REGISTRO.

BASE DE DATOS

17.850 pts.

NEWWORD

Programa de tratamiento de textos mejorando todo lo anterior. Manual y programa en español, que le enseñarán con facilidad y rapidez lo más avanzado en procesadores de textos. Compatibilidad funcional con WORDSTAR incluyendo muchas capacidades adicionales.

Tiene un potente MAIL-MERGE con opción de selección de destinatarios por criterios base de datos, creación de documentos, impresión de etiquetas. Utiliza todo el espacio de disco. Ensamblaje de textos, sustitución, etc., de la forma más fácil: autohace copias de seguridad. ¡NUNCA PERDERA UN TEXTO!

- Ñ, ACENTOS, DIERESIS, ETC...
- PRESENTACION EXACTA EN PANTALLA DEL FUTURO DOCUMENTO IMPRESO.
- INTERCAMBIOS DE FICHEROS CON CRACKER.
- VARIABLES SUSTITUIBLES EN IMPRESORA.
- POTENTE CALCULADORA.
- COMPROBADOR ORTOGRAFICO Y GRAN DICCIONARIO (45.000 TERMINOS AMPLIABLES).
- POSIBILIDAD DE LECTURA DE FICHEROS DE DELTA, CARD BOX, SUPERCALC, DBASE II, ETC...

TRATAMIENTO DE TEXTOS

17.850 pts.

CRACKER II

El CRACK de las hojas de cálculo, la que deja detrás al resto. Funciones nunca vistas, formateo de fechas, salvaguardia continua sobre un fichero. Realiza automáticamente copias de seguridad. Además de las tradicionales funciones, CRACKER II posee funciones lógicas, estadísticas y de alta matemática. Intercambia datos con NEWWORD, bases de datos y la mayoría de las hojas de cálculo.

- CELDAS PROGRAMABLES.
- FUNCIONES ESPECIALES: Fecha, días; desde y hasta la fecha de la semana, del año, lapso de tiempo, retraso, beep entrada, saludo usuario.
- SISTEMA DE AYUDA ON-LINE.
- SUMA CONDICIONAL.
- TOMAR DECISIONES EN LA HOJA.
- 18 MODOS GRAFICOS DISTINTOS.
- TRADICIONALES FUNCIONES MATEMATICAS Y AMPLIACION, FUNCIONES ESTADISTICAS Y LOGICAS.
- GENERA GRAFICOS EN BASE A LOS DATOS.

HOJA DE CALCULO

17.850 pts.

EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA





Informática

y estas son sus estrellas.

NUCLEUS

NUCLEUS más que una estrella una constelación; tres ESTRELLAS en un SUPERPROGRAMA, la solución a cualquier aplicación por compleja que sea, NUCLEUS es GENERADOR DE PROGRAMAS, BASE DE DATOS Y GENERADOR DE INFORMES.

Toda la información es multi-intercambiable y de libre acceso por cualquiera de los demás programas. Así los datos de la base los condicionamos y utilizamos en el generador de programas y los imprimimos a través del generador de informes.

- *GENERADOR DE PROGRAMAS EN MALLARD BASIC.*
- *CREACION DE BASES DE DATOS RELACIONALES.*
- *GENERADOR DE INFORMES.*
- *DISEÑADOR DE FORMATOS.*
- *DISEÑADOR DE PANTALLAS.*
- *CODIGO FUENTE DE LIBRE ACCESO Y LIBRE DE ERROR.*
- *DISEÑA SU PROPIO SISTEMA.*
- *MAILMERGE.*

GENERADOR DE PROGRAMAS

26.780 pts.

BRAINSTORM

La revolución del pensamiento, BRAINSTORM es un programa que piensa con Vd.

El compañero ideal para el empresario, director o cualquier persona que tenga que planificarse o tomar decisiones.

BRAINSTORM es la ayuda necesaria para su organización. El programa que se ha standarizado en Inglaterra, tan necesario, útil y popular como una base de datos o un tratamiento de textos.

- *ORGANIZA POR RANGOS.*
- *ACCESO DESCENDENTE PORMENORIZADO.*
- *PLANIFICACION A NIVEL DIA.*
- *DECISIONES A LARGO PLAZO.*
- *REVISION DE PROBLEMAS.*
- *SIMULTANEIZACION DE TAREAS.*
- *PROCESO TOP/DOWN.*

ORGANIZADOR DE IDEAS

17.850 pts.

STARCOM

Piii... su ordenador le comunica:

La revolución de las comunicaciones, de la mano de OFITES INFORMATICA, llega a España. El nuevo mundo de las comunicaciones digitales lo tiene a su disposición, las redes de transmisión electrónica digitalizada, con su PCW 8256 o PCW 8512 a través de un interface RS 232-C con otros ordenadores, redes de transmisión de datos, etc..., Vd. podrá enviar o recibir ficheros de texto o de datos, ASCII, etc..., creados por NEWWORD y otros...

- *TRANSICIONES DIRECTAS EN RED.*
- *COMPATIBILIDAD CON NEWWORD.*
- *POSIBILIDADES DE TRANSMISIONES VIA MODEM, RED TELEFONICA.*
- *COMUNICACION INSTANTANEA.*

COMUNICACIONES

17.850 pts.

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:



Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

CINCO PULGADAS UN CUARTO PARA AMSTRAD

Mucha es la espectación que tenía la llegada de las unidades de disco de 5'' 1/4, para los ordenadores CPC 6128 y CPC 464. Una de las primeras en entrar en nuestro mercado ha sido la realizada por la casa inglesa Cumana, ya conocidas en este país por sus unidades para el Spectrum y para el QL. Una unidad de disco extraplana, bastante estética, como es de costumbre en esta casa, y bien acabada exteriormente. En una unidad de cinco pulgadas y cuatro de simple cara y simple densidad.

La unidad de disco está provista de un cable de conexión para el ordenador. No necesita de ningún tipo de interface. El cable viene provisto de dos conectores, uno en su extremo, y el otro en la parte central del mismo. Este último viene preparado para introducirlo en la salida del Bus de expansión, el otro conector está preparado para en su día conectarle una tercera unidad de disco.

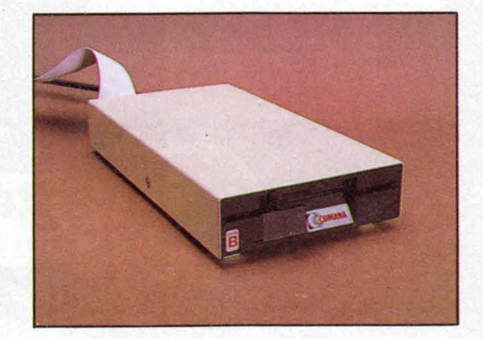
Empezamos a trabajar

Una vez conectada, empezemos a trabajar con la unidad de disco. Lo primero que hacemos es comprobar si el sistema de nuestro ordenador le acepta, para lo cual mandamos el control de la misma a la unidad B (mediante la orden de AMSDOS IB). En el momento y después de efectuarse su puesta a punto nos remite el mensaje de READY.

A la hora de empezar formatearemos un disco de 5'' 1/4 para empezar a trabajar. Aquí nos planteamos la primera incógnita, al intentar formatear los discos con mayor capacidad. Intruducimos el CP/M y una vez dentro del sistema cargamos el programa Disckit3. Como todos sabemos el programa Disckit3, es el encargado de formatear discos vírgenes, copiar programas para copias de seguridad y verificar los discos. Al cargar el sistema nos dimos cuenta de que aceptaba la segunda unidad de disco, al salir en el mensaje de presentación dos unidades conectadas. Empezamos a formatear el disco pulsando la opción correspondiente. En la pantalla nos irá reflejando el número de pistas que estábamos formateando, esperando que al llegar a treinta y nueve continuase. Pero desgraciadamente no fue así, al llegar a la número 39 (pista cuarenta al contarse el cero como primera), se detuvo el formateo y nos indicó el mensaje de operación concluida.

Seguidamente nos salimos del programa y también del sistema operativo mediante AMSDOS. A continuación para saber la capacidad que teníamos en el nuevo disco efectuamos un catálogo del mismo para ver cuál era su capacidad realmente, el resultado fue el mismo que si de un disco de tres pulgadas se tratara. Teníamos un total de 179 kbytes libres; memoria libre en el disco.

El siguiente paso a seguir es comprobar si funciona bien salvar y cargar programas. Empezaremos por probar programas en Basic, luego programas salvados en ASCII, Basic Protegido y finalmente programas en código máquina (programas en binario).



Grabar un programa

Haremos un pequeño programa que contenga bucles, impresión en pantalla y algunas llamadas a máquina. Una vez realizado seguimos los pasos normales para su almacenamiento en un disco. Primeramente direccionamos la unidad receptora mediante IB. Seguidamente nos disponemos a salvarlo con la orden de Basic:

SAVE «Nombre. Extensión»

El almacenamiento se realiza normal, y pasamos a comprobar que está bien almacenado. Para ello reseteamos el equipo y efectuamos un catálogo del disco B y posteriormente la carga del programa. Cuando se carga hacemos un listado y comprobamos que lo ha realizado todo correctamente. Sólo nos falta ejecutarlo por si acaso al cargarlo los punteros se han perdido u otro problema por el estilo. El resultado es satisfactorio, el programa se ha ejecutado perfectamente. Ahora probamos si puede trabajar con programas almacenados de tipo ASCII. Utilizamos el mismo programa pero salvado con ASCII:

SAVE «Nombre.Bas»,a

El almacenamiento como era de esperar es un poco más lento que si de un programa en Basic se tratara. Comprobamos si se encuentra en directorio y vemos que sí. La carga y ejecución posterior fue normal, con lo que aseguramos que puede cargar y almacenar programas en el disco de 5'' 1/4 como si fuera uno de tres pulgadas.

Con un programa en Basic protegido el resultado fue exactamente igual que con los casos anteriores, todo se realizó y resultó de una forma normal. Sólo nos queda la cuestión de probar con programas en código máquina.

Una forma fácil de comprobarlo es realizar una pantalla y salvarla al disco y posteriormente ejecutarla. Eso fue lo que hicimos y los resultados fueron de un auténtico éxito.

Para ver si podían introducirse ficheros, utilizamos un programa de contabilidad que pudieran estar los datos en la segunda unidad, y funcionó sin problemas.

En definitiva, la gestión de la unidad de 5'' 1/4 es idéntica a una segunda unidad de tres pulgadas. Se puede realizar todo tipo de almacenamiento en el disco y su posterior

carga a la memoria. Los comandos de AMSDOS referentes a borrar, catálogo, etc. funcionan correctamente.

Utilización del CP/M con la Unidad.

Cuando probamos la unidad trabajando en CP/M 2.2 o CP/M 3.0 (PLUS), lo primero que necesitábamos era tener un par de discos de cinco pulgadas formateados en los respectivos sistemas. Para ello utilizamos el programa Disckit2 o Disckit3, dependiendo del sistema deseado. También se puede realizar con el Disckit3, y a la hora de pedir la introducción del disco con sistema, introducir primeramente uno y luego el segundo sistema.

Cuando hayamos realizado el **«formateo»** introduciremos en cada uno todo los programas y utilidades de cada sistema.

Para introducir los programas podemos utilizar bien el PIP.COM o el propio Disckit.

Una vez que tengamos todos los programas y utilidades traspasados, empezaremos a trabajar con ellos. Empezaremos por utilizar el CP/M 2.2.

Introducimos el programa ED.COM para comprobar si funciona el editor de textos. Lo cargamos y empezamos a trabajar con él. Utilizamos todos los comandos con toda normalidad. A continuación metemos el DDT.COM con un programa de texto en pantalla. Cuando fuimos a introducir el programa para depurar hay que decir que le costó un poco de trabajo cargarlo, tal vez por una mala grabación del mismo. Una vez cargado todo funcionó bien.

Así continuamos con los programas de más utilización y el resultado fue idéntico a los anteriores, todo iba correctamente.

Con el CP/M Plus nos costó un poco de trabajo introducirlo.

A la tercera vez, conseguimos que nos aceptara la segunda unidad de disco. Esto se comprueba en el mensaje de presentación, en él nos indica la TPA, las unidades conectadas, etc.

Para ver el traslado de programas de una unidad a otra utilizamos el comando PIP. Con este modificamos algunos programas de un directorio a otro.

Para probar alguna salida diferente de la consola o la unidad de disco, pensamos en mandar un fichero desde la segunda unidad a la impresora.

Cargamos el programa PIP.COM y lo preparamos para mandar un fichero ASCII a la impresora. Si funciona, podemos asegurar que está la unidad perfectamente adaptada a todas las salidas y entradas de una unidad de discos.

El fichero a los pocos segundos fue impreso en la impresora, completando así la prueba de salida.

Trabajando con AMSDOS

Como todos sabemos el AMSDOS son los comandos que incorporan a nuestro Basic después del signo |. Vamos a probar uno a uno todos los comandos para ver cómo se comporta la unidad de disco.

|A; |B

Con este comando nos movemos de una unidad a otra. Al introducirlo el control se nos marcha de una a otra unidad, funciona.

|CPM

Esta orden la hemos probado antes a la hora de introducir el CP/M, y comprobamos que éste se carga desde la unidad A, pudiéndose más tarde trabajar con la B.

|DIR

Esta orden nos mostró todo el directorio del disco, situándonos antes en la segunda unidad, a la vez que probamos sus parámetros.

|DISC; |DISC.IN; |DISC.OUT

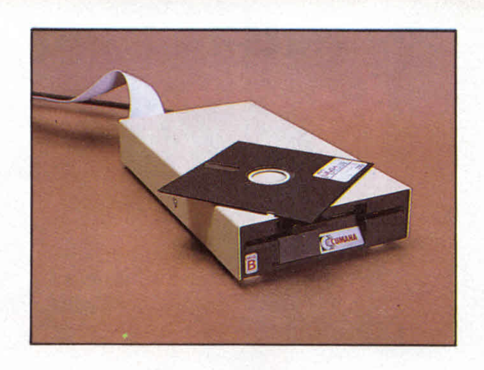
La primera orden equivale a las dos siguientes, y nos dice que todas las operaciones de entrada y salida son con el disco.

|DRIVE

Nos cambia la unidad de disco implícita. Es una forma de saber si tenemos bien conectada la segunda unidad, de otra forma esta orden fallaría por no encontrar la segunda unidad.

|ERA

Pudimos comprobar que borra todos los ficheros de la segunda unidad como de la primera. También se probó con sus parámetros y trabajó normalmente.



|REN

Renombramos algunos de los programas para utilizar diferentes parámetros con el |ERA y |DIR.

|USER

Escribimos algunos usuarios en el disco, e introducimos algunos programas en los mismos. Luego pudimos acceder a ellos colocándonos previamente en el usuario conveniente.

Problemas encontrados.

El primero de ellos es la falta de manual. Aunque no sea imprescindible para su instalación, sí es conveniente para su utilización y mayor aprovechamiento.

Cuando buscamos un sitio para su colocación, pensamos en situarlo en la parte derecha del ordenador, pero debido al corto cable de expansión la colocación de la unidad disco es un poco dificultosa, ya que al ser el conector central el que se introduce en el bus de expansión, obliga a colocar dicha unidad perpendicular a la consola del ordenador y detrás de ésta. Si por casualidad tenemos un monitor a color, el problema se acentúa aún más debido al mayor tamaño de éste, que debe ser desplazado hacia un lateral, preferentemente a la izquierda con la consiguiente incomodidad para el cable de la impresora en caso de tenerla. El resto del cable de conexión de la unidad de discos queda temporalmente obsoleto, siendo incómodo además de poco estético al quedarse suelto por la mesa. El conjunto estético que nos supone el utilizar esta unidad es pues bastante lamentable.

Otro inconveniente es el ruido que produce la unidad de disco al trabajar. Este ruido, se asemeja al que produciría una unidad de disco normal si tuviera suelta la cabeza o no tuviera engrasado convenientemente el motor.

Además la velocidad a la que trabaja esta unidad es inferior a la que trabaja la unidad de disco de tres pulgadas.

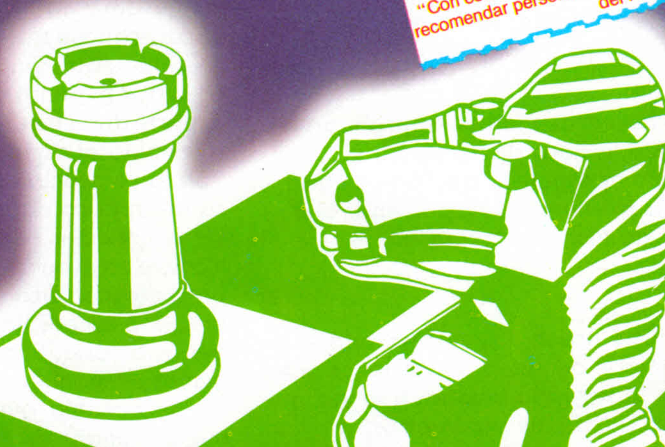
¡NOVEDAD MUNDIAL!

AMSTRAD

1er JUEGO PCW-8256

3-D CLOCK CHESS

OPINA
Jon Speelman
Maestro Internacional (Campeón Británico - 1985)
"Con este programa de juego y los excelentes gráficos 3-D, puedo recomendar personalmente 3-D CLOCK CHESS a todos los amantes del Rey de los Juegos"



PRINCIPALES CARACTERISTICAS:

POTENTE: Reúne las últimas técnicas desarrolladas para los programas de ajedrez basadas en la inteligencia artificial.

RAPIDO: 3-D CLOCK CHESS es increiblemente rápido, y puedes determinar los niveles de juego seleccionando el tiempo de respuesta en cualquiera de los cuatro modos de juego, lo que te da cientos de posibles niveles.

ANALITICO: Este programa tiene una fuerza insospechada: en los niveles de torneo el ordenador se plantea hasta siete movimientos consecutivos eligiendo literalmente entre decenas de miles de líneas de juego.

AMISTOSO: Posee todas las posibilidades que un verdadero jugador de ajedrez puede necesitar, llevándote totalmente informado acerca de su propio proceso pensante. Es un profesor ideal para el principiante y un tutor ideal para el experto.

SUPERGRAFICOS 3-D: Los últimos desarrollos en visión tridimensional y bidireccional del tablero y piezas del ajedrez, así como de la exclusiva opción del Torneo Cronometrado de Ajedrez, con la interesante posibilidad de "jugadas en tiempo compensado"

P.V.P. 3.950 pts. + I.V.A.

Producido en exclusiva para España por:
ACE DISTRIBUCION, S.A.
(c) CP. SOFTWARE

Distribuido por:
microbyte
P.º Castellana, 179, 1.º
Tel.: (91) 442 54 44.
28001 MADRID

ACE DISTRIBUCION
(Cataluña y Baleares)
c/ Tarragona, 110-112.
Tel.: (93) 325 10 58.
02015 BARCELONA
Telex: 93133 ACEEE

MICROCOMPUTADORAS DICCIONARIO DE TERMINOS

A la hora de saber lo que significan muchos términos en informática, nos encontramos con muchos problemas, ya que la traducción literal que encontramos en cualquier diccionario, no se asemeja en nada a lo que esperábamos. Para esto nos es necesario un diccionario especializado, como el que nos presenta Mac Graw-Hill.



Un diccionario que nos aportará información sobre lo que es un byte, bit, nibble y todo tipo de palabras. Además podremos buscar todo tipo de términos técnicos como que es una memoria de burbuja, un interface de entrada salida, una biblioteca de programas, etc.

Es un diccionario que se extiende lo suficiente en todas sus explicaciones, que nos da respuesta a cualquier tipo de información que necesitemos sobre microcomputadoras, ya sean cuestiones electrónicas, informáticas, físicas, o históricas, fechas, datos sobre personajes, etc.

Todos los términos se tratan alfabéticamente como si de una sola palabra se tratase.

Utiliza un sistema de referencias cruzadas para ayudar a buscar términos que pudieran localizarse en varios lugares del diccionario.

La cabecera de cada página contiene la primera y última palabra que contiene.

El único inconveniente es que las referencias deberemos hacerlas en inglés, sí en inglés, aunque posteriormente nos aparezca la traducción y la explicación en nuestro idioma, así que si desconocemos la traducción al inglés de lo que deseamos buscar no será una tarea fácil.



POGRAMACION AVANZADA DE AMSTRAD

Un libro basado en el CPC464, al que en su prólogo compara con un iceberg, por la parte que permanece escondida, el libro en sí es uno más de los editados para este ordenador, pero el tema tratado le hace diferente. El objetivo principal es mostrarnos cómo manejar los periféricos con el **Amstrad**.

El libro deja bien dicho en su prólogo, que hay que poseer «literatura» sobre el Z-80, ya que se hacen continuas referencias a subrutinas de máquina para este procesador.

El libro se divide en tres partes:

La primera está dedicada a las entradas, empezando a hablarnos del sistema solapado de memoria someramente como puede manejar la Z-80 64 K de RAM y 32 K de ROM.



A continuación empieza a contarnos cosas sobre la estructura interna de la máquina. Como funciona su sistema de memoria, hablando de ,los periféricos internos, controlador de teclado, los diversos dispositivos de visualización, el controlador de sonido, etc. Suministrando un completo mapa de las direcciones de entrada/salida.

Nos muestra también cómo trabaja el cassette, con una completa descripción del bloque de cabecera del mismo, también nos habla sobre el teclado, nos cuenta para qué sirven las distintas direcciones del área RST, concluye el capítulo con una reducida guía de llamadas a las subrutinas del firmware.

El segundo capítulo está dedicado al interfaz, hablando muy escasamente del puerto de expansión, tan sólo tres páginas.

Y en el tercer capítulo trata con gran extensión el tema de las salidas, explicando cómo construir una expansión de la tarjeta madre, contando cosas sobre el interfaz paralelo, cómo crear un punto de impresora alternativo, el interfaz serie, y finaliza hablando sobre las ROM auxiliares, que son, como tipos, formato y aplicaciones.

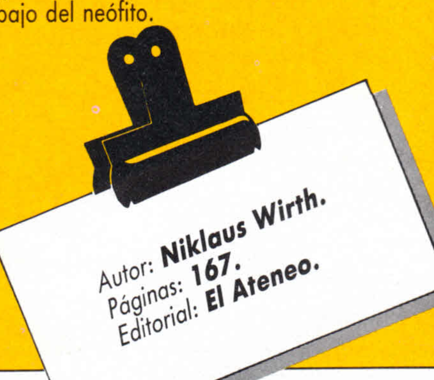
Al final del libro encontramos un programa que nos permitirá visualizar o imprimir el contenido de la RAM y de la ROM.



En definitiva un libro que es muy aconsejable para el programa avanzado de código máquina.

INTRODUCCION A LA PROGRAMACION SISTEMATICA

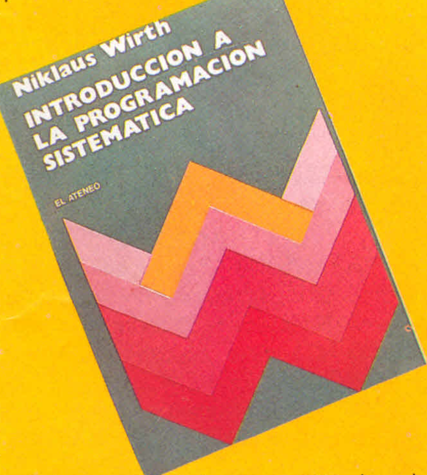
El libro se sitúa en los comienzos del Pascal, aunque en el libro se le menciona como una extensión del ALGOL-60. El título hace referencia principalmente a los seis primeros capítulos, esbozando el funcionamiento de una computadora y con temas que ayudan a sistematizar y verificar el trabajo del neófito.



Libros

A partir del capítulo 7 se estudian los métodos recursivos e iterativos; se muestra la potencia de los tipos, el almacenamiento en ficheros secuenciales, manejo de punteros, declaraciones de procedimientos y funciones, etc., que constituyen toda una demostración de la potencia del lenguaje.

Cabe resaltar la separación y explicación en los tipos de sustitución y asignaciones generales. El manejo de las mismas obliga a conocer muy bien las prestaciones de nuestro compilador. Sin embargo, el libro no responde a lo que es un manual de compilador, pero puede ayudar a simplificar el manejo de un compilador determinado.



Su lectura proporciona una visión clara de la descripción de identificadores. Todos los conceptos y sentencias están muy bien explicados y se hacen consideraciones sobre el ámbito de validez de los mismos.

Se echa de menos una más clara separación entre las partes declarativa y ejecutable. La parte dedicada a ENTRADA-SALIDA es muy superficial debido a que varía mucho de un compilador a otro.

PROGRAMACION EN C: INTRODUCCION Y CONCEPTOS AVANZADOS



Como indica su título, es un libro muy completo. Parte de la base que el lector apenas tiene conocimientos de programación. Los autores son los creadores de la versión C para el IBM PC, el Lattice C. Esto no quiere decir que vaya dirigido hacia dichos usuarios; se extiende sobre su aplicación en sistemas operativos como el XENIX, UNIX, etc.

Se empieza con una introducción al C, procedencia, utilidad manejo y, ¡cómo no!, cualidades. Para regocijo del novicio se dedica un capítulo a la descripción de datos, variables, constantes y literales el forma en que son tratados por C.

A continuación nos introducimos en una típica iniciación de C: programas de impresión de cadenas, manejo de los distintos tipos de caracteres y en el uso de la palabra clave DEFINE #. También se dedica información cautelosa y amplia sobre operadores, entrada/salida, condiciones, bucles, funciones, etc.

Se hace especial hincapié a lo largo de un capítulo sobre la potencia del preprocesador y de la biblioteca disponible.

Concluye con una serie de apéndices que serán de gran utilidad al programador a la hora de trabajar con este lenguaje.

Todas las explicaciones son muy claras y los autores tratan el texto con cordialidad, bromeando ampliamente a lo largo del mismo.



CP/M: GUIA DEL USUARIO

Un libro provechoso para sacarle todo el partido al sistema operativo CP/M. A través de sus páginas iremos viendo los diversos comandos y órdenes transitorias que nos brinda CP/M, con una completa explicación de las mismas.

Se divide en ocho capítulos y otros tantos apéndices en los cuales se describe la ruta a seguir en los senderos CP/M.



Los dos primeros capítulos tratan los antecedentes históricos del sistema así como sus órdenes. Las opciones con la tecla CTRL también son expuestas.

Poco a poco, nos vamos adentrando en el sistema operativo. Ordenes transitorias, exhaustivas explicaciones del editor y ensamblador; eso sí, desde un punto de vista un tanto triunfalista...

Avanzando más en el manejo del sistema operativo encontramos el método para enfrentarnos a archivos transitorios y sistemas de multiproceso. Lo adecuado para estar al corriente de las técnicas profesionales.

Los últimos capítulos se dedican a la descripción técnica del mismo para aquél que desea hacer sus pinitos bajo los auspicios del sistema.

Completan la descripción una guía rápida, código ASCII, comparación entre distintas versiones, bibliografía, etc., todo ello incluido en los apéndices del mismo.



LISP O EL DOMINIO DE LA IA

Autor: *R. Garrote*

(DE EQUAL (SEX1 SEX2)
(COND ((ATOM SEX1) (EQ SEX1 SEX2))
((ATOM SEX2) NIL)
((EQUAL (CAR SEX1) (CAR SEX2))
(EQUAL (CDR SEX1) (CDR SEX2)))
(T NIL)
))

Cuando vi por primera vez un programa LISP quedé sorprendido. ¿Cómo era posible que eso, que no tenía asignaciones, ni bucles, ni saltos, pudiese hacer algo?

Para cargar tu intérprete de LISP teclea RUN "MINILISP". Este programa es el cargador: te muestra la insignia del programa y te pide el tamaño de la tabla de identificadores, que es el diccionario interno del intérprete. Debes dar un número entre 100 y 1.000. Pongamos, de momento, 300. (El programa está diseñado de modo que te deje la mayor cantidad posible de memoria libre —esto lo consigue con el comando CHAIN MERGE y su opción DELETE— de modo que el programa se carga en cuatro etapas. Entre estas etapas hace dos pausas pronunciadas para inicializar las estructuras internas). **Cuando el intérprete de MINILISP esté dispuesto escucharás un pitido y aparecerá en pantalla un mensaje indicándote la cantidad de memoria disponible.** Justo debajo aparece el «saludo» de tu intérprete:

>

Uno se puede entonces imaginar que dentro de la máquina se halla un geniecillo que está dispuesto a satisfacer todas tus órdenes siempre que se las des en su idioma, que es LISP. (El geniecillo de nuestro computador es un poco perezoso para cumplir los recados que le pedimos).

Vamos a pedirle al genio algo que tanto tú como él entenderéis:

> (+ 2 2) (¡y ENTER claro!)

En efecto, queremos que el genio nos diga cuántas son dos más dos. Cuando el genio comprende nuestra solicitud indica con puntos suspensivos que está pensando (evaluando, se dice). Cuando sabe la respuesta nos enseña su reloj para que veamos cuánto ha tardado en calcularla y nos dice que dos más dos son cuatro. ¡Magnífico!

El intérprete de LISP «lee» nuestro deseo, lo evalúa, imprime el resultado y nos indica a continuación que está dispuesto a cumplir un nuevo mandato.

¿Cómo podemos comunicarnos con el duende de LISP y de qué forma nos responderá? En principio se le pueden pedir cosas sencillas, pero escritas en una forma un tanto extraña (a los que hayáis programado algunas calculadoras tal vez os suene: se llama notación polaca). Por ejemplo. Para que calcule (2*3)+15—(6/2) se podría escribir:

> (+ (* 2 3)
> (SUB 15
> (DIV 6 2)))

¡No parece LISP un lenguaje especialmente pensado para realizar operaciones aritméticas! Además, MINILISP opera sólo con números enteros. LISP es un lenguaje especializado en manipular listas de objetos (el nombre de LISP viene de LISt Processing, procesamiento de listas). Los elementos básicos que maneja LISP son los átomos.

Pero, ¿qué es un átomo? En principio, un átomo es cualquier sucesión de letras y números que empiece por una letra. Por ejemplo, son átomos

ATOMO átomo Casa A123 Lista LIstA NIL T

y no son átomos

123a (EstoNoEsUnAtomo)

Tu intérprete de LISP no distingue entre mayúsculas y minúsculas, de modo que ATOMO, Atomo y AtoMO son uno y el mismo objeto.

Sin embargo, también son átomos

ANDRES PEREZ ES UNA ? *ESTRELLAS*

Todo estos átomos se llaman átomos literales. Podemos dar entonces una definición más general de los átomos literales: un átomo literal es cualquier sucesión de símbolos con dos únicas restricciones:

i) Un átomo literal no puede empezar ni por un dígito ni por el símbolo '.

ii) Un átomo literal no puede contener ninguno de los siguientes símbolos: ( ) . %

Las razones de que estos símbolos no se puedan utilizar las iremos explicando poco a poco. Ya hemos visto que LISP también maneja números. Los números también son átomos aunque, no son átomos literales. (A veces se llaman átomos numerales o numéricos).

En LISP, todas las instrucciones son llamadas a funciones (véase el artículo sobre Inteligencia Artificial en el número xxx). Una función básica (primitiva) en LISP es QUOTE. Esta función tiene como valor su argumento, es decir

> (QUOTE ANTONIO)

vale ANTONIO. QUOTE tiene como misión impedir que se evalúe su argumento. Por ejemplo, cuando tú escribes en Basic.

PRINT 2+2

el intérprete de Basic evalúa la expresión 2+2 y escribe su valor: 4. Si quieres que no evalúe, debes escribir

PRINT "2+2"

De la misma forma

PRINT ANTONIO

escribiría el valor de la variable ANTONIO, mientras que

PRINT "ANTONIO"

escribiría ANTONIO. Para impedir que se interprete ANTONIO como una variable debes entrecomillarlo; ese efecto de **«entrecomillado»** es el que se consigue en LISP con la función QUOTE.

En LISP, los átomos literales no tienen valor excepto los átomos NIL y T cuyos valores son respectivamente NIL y T. Este par de valores se usa con frecuencia como valores booleanos: NIL equivale a falso y T a cierto (True, en inglés). En MINILISP, existen otros dos átomos literales con valor: PIZQ que vale (, cuyo valor es). Sirve para que puedas escribir ambos paréntesis. Los átomos numerales si tienen valor y su valor es el número que representan. Si se evalúa un átomo que no tiene valor se produce un error. (Algunos átomos literales pueden actuar como variables y guardar temporalmente valores, pero de las variables hablaremos luego).

Como hay que utilizar mucho la función QUOTE se ha ideado una abreviatura para ella (igual que puede utilizar ? en vez de PRINT cuando programas en Basic). Esta abreviatura es '. Por tanto

también vale ANTONIO. En adelante, en lugar de QUOTE utilizaremos su abreviatura.

Otra función primitiva de LISP es CONS, que produce pares de objetos. Por ejemplo:

> (CONS 'A 'B)

vale (A.B). Nos encontramos entonces ante un objeto que no es un átomo. Se llama par, el par que tiene como primer elemento A y como segundo elemento B. ¿Por qué hemos utilizado QUOTE? Pues porque CONS también evalúa sus argumentos y queríamos obtener el par formado por A y B y no por sus valores. Un ejemplo más:

> (CONS 'A 1)

vale (A. 1). El átomo 1 tiene como valor 1 y por eso no hay que ponerle QUOTE delante.



Ahora hagamos la operación contraria: dado un par, obtener sus componentes. Hay dos funciones que hacen esto: CAR devuelve el primer elemento del par y CDR su segundo elemento:

>(CAR ' (A.B))

>(CDR ' (A.B))

vale B. Además de construir y «destruir» objetos podemos hacer preguntar al genio sobre estos objetos. Por ejemplo, el predicado EQ se utiliza para comprobar la igualdad de dos átomos literales mientras que para comprobar la igualdad de números debe usarse=

>(=4 8)

vale NIL, es decir, es falso, mientras que

>(EQ ' A (CAR ' (A.B)))

vale T. Date cuenta que el segundo parámetro de EQ, (CAR ' (A.B)), no lleva QUOTE. En consecuencia, el genio evalúa esto para dar A, según vimos en un ejemplo anterior. El resultado de evaluar 'A también es A y por tanto se verifica la igualdad.

Vamos a construir ahora estructuras más complicadas.

> CONS 'ANA (CONS 'ES 'BONITA))

vale (ANA. (ES.BONITA)). CONS sólo admite dos parámetros, de modo que para obtener estructuras de más de dos elementos debe formarse pares de dos en dos. Para reflejar esta forma de construir estructuras es mejor escribir

>(CONS 'ANA
> 'ES
> 'BONITA)))

Si ahora le preguntamos al genio

+(CONS 'A NIL)

nos responderá con (A). *¿Qué ha ocurrido?* ¿No debería ser la respuesta (A. NIL)? Antes dije que LISP es un lenguaje especializado en manipular listas y, sin embargo, hasta ahora sólo conocíamos átomos y pares. Tanto átomos, como pares y listas se conocen con el nombre genérico de expresiones simbólicas (S-expresiones o SEXos). Ocurre que el átomo NIL es un poco «raro». Pero antes de seguir daré la definición de lista:

i) () es una lista con 0 elementos (se llama lista vacía).

ii) el par que tiene como primer elemento una S-expresión S y como segundo elemento una lista L es, a su vez, una lista que tiene como primer elemento la S-expresión S y como resto de la lista la lista L.

Para los que no estéis acostumbrados a las definiciones recursivas —las que usan el objeto que se pretende definir en la propia definición— daré otra definición: una lista es o una lista que no tiene elementos, representada por (), o cualquier cosa que contenga átomos, pares u otras listas entre un paréntesis abierto "(" y un paréntesis cerrado ")". Por ejemplo:

(ESTO ES UNA LISTA DE 7 ELEMENTOS)

(ESTO ES (UNA (LISTA DE 3) ELEMENTOS))
(ESTO (TAMBIEN.ES) ((UNA)) LISTA)

El tercer ejemplo es una lista de tres elementos, el tercero de los cuales es a su vez una lista de tres elementos.

(UNA (LISTA DE 3) ELEMENTOS)

y cuyo segundo elemento es otra lista de tres elementos.

(LISTA DE 3)

Veamos algunos ejemplos de cosas que no son listas·

LE FALTAN LOS PARENTESIS
((LE FALTA) EL PARENTESIS DERECHO

Todos los objetos que manipula LISP o son átomos, o son pares, o son listas. Sin embargo, existe en LISP un objeto un tanto esquizofrénico: es un átomo que también es una lista o una lista que también es un átomo. Tiene dos representaciones: como átomo, NIL, y como lista () (por eso a veces se dice que es BISE-Xual). LISP no distingue entre una y otra representación: para LISP son una y la misma cosa. En adelante, usaré la representación que me parezca más conveniente, sin hacer referencia a la otra.

Por la definición de lista, como A es un átomo y NIL es una lista, entonces (CONS 'A NIL) también es una lista, que tiene como primer elemento A y como resto la lista vacía (NIL o ()). ¿Entiendes ahora por qué (CONS 'A NIL)

es (A)? Para construir una lista sólo tiene que poner como último parámetro del CONS «más interno» () o NIL. Por ejemplo:

> **(CONS ' ESTO**
> **(CONS 'ES**
> **(CONS ' UNA**
> **(CONS 'LARGA**
+ **(CONS ' LISTA**
+**)))))**

cuyo valor es (ESTO ES UNA LARGA LISTA). En este caso, el CONS más interno es (CONS ' LISTA ()).

Construir listas de esta forma es un duro trabajo, de modo que existe en LISP una función que puede tener cualquier número de argumentos y que produce una lista con los resultados de evaluar cada uno de sus argumentos; es la función LIST.

> **(LIST ' ESTO ' ES ' UNA ' LARGA ' LISTA)**
vale (ESTO ES UNA LARGA LISTA).

Ya habrás observado la cantidad de paréntesis que hay que escribir en LISP. Esto no suele ser un problema salvo cuando hay que cerrar todos los paréntesis que permanecían abiertos. Para no tener que andar contando con cuidado, en MICROLISP puedes escribir el símbolo / para indicarle al genio que quieres cerrar todos los paréntesis.

Hasta ahora no hemos necesitado conocer las propiedades de los objetos que le presentábamos a LISP pues ya sabemos que 1 es un numeral, PERRO es un átomo y (JUAN.MARI) es un par. Sin embargo, vamos a empezar a tratar con variables (sí, ¡por fin!) y entonces sólo conoceremos el nombre del objeto y no sus características. Por eso voy a explicar algunas funciones, llamadas reconocedores, que tienen como fin informarnos de las características de los objetos que manejemos. ATOM nos dice si un SEXo es un átomo o no

> **(ATOM 'A)**
vale T.
> **(ATOM 1)**
vale T.
> **(ATOM (CAR (A.B** /
vale T, pues (CAR (A.B)) vale A, que es un átomo. Pero
> **(ATOM ' (CAR (A.B** /
vale NIL, pues (CAR (A.B)) es una lista. Esta es una característica muy importante de los programas LISP: todo programa desde otros programas LISP y evaluarios inmediatamente. Esta es una de las ventajas de LISP sobre otros lenguajes de programación y una de las razones de que se utilice en IA (Inteligencia Artificial). Más adelante veremos ejemplos de esto.

Otros reconocedores son LITATOM, NUMBERP y PAIRP. El primero nos dice si un objeto es un átomo literal, el segundo si es un número y el tercero si es un par.

> **(LITATOM ' A)**
vale T, pero
> **(LITATOM 1)**
vale NIL.
> **(NUMBERP 1)**
vale T y
> **(NUMBERP 'A)**
vale NIL.
> **(PAIRP ' (A B C))**
vale T, pues (A B C) es la representación en forma de lista de (A. (B. (C.NIL))), que es un par con primer elemento A y segundo elemento (B. (C.NIL)).

La función NULL sirve para reconocer la lista vacía.

> **(NULL ())**
vale T. Espero que sepas cuanto vale.
> **(NULL NIL)**
Si no estás muy seguro pregúntaselo a tu genio. Un ejemplo más.
> **(NULL ' (A B))**
vale NIL.

Hasta aquí hemos visto algunas de las funciones internas del sistema. Ahora veremos cómo podemos definir las nuestras. Supongamos que estamos escribiendo un programa para encontrar la pareja ideal de algunas personas y cuando la encontramos se casan. Para casar a la feliz PAReja podríamos definir:

+ **(DE BODA (JOSE ANA) (CONS JOSE ANA))**

¿Por qué ahora JOSE y ANA no llevan QUOTE? Pues porque ahora JOSE y ANA actúan como variables (se llaman parámetros de la función) y no como objetos. Podríamos haberles llamado X e Y, pero una boda entre X e Y puede sonar un tanto «robótica». En LISP, cualquier átomo literal sirve como nombre de función o como nombre de variable, pero no puedes definir funciones con los nombres de las palabras reservadas de LISP. (Si quieres saber cuáles son estas palabras reservadas dile a tu intérprete (OBLIST). Esta función te muestra en pantalla todos los identificadores que MINILISP conoce hasta el momento).

La forma de definir funciones en LISP es semejante a como se hace en Basic: se utiliza una palabra reservada para indicar que lo que sigue es una definición, DE, y luego se da el nombre de la función a definir, BODA, seguido de los parámetros de la función, JOSE y ANA, y de las instrucciones para calcular el valor de la función, (CONS JOSE ANA).

Ahora podemos utilizar la función BODA que hemos definido de la misma forma que una función interna del sistema. Al usarla, deberemos dar valores a los parámetros. Por ejemplo.

+ **(BODA ' SEGISMUNDO ' ROSALINDA)**

y el feliz enlace sería (SEGISMUNDO. ROSALINDA).

La posibilidad de definir funciones nos permite modificar los nombres de las funciones del sistema LISP. Por ejemplo, las funciones CAR y CDR pueden tener nombres adecuados cuando se trata de obtener las componentes primera y segunda de un par, pero

> **(CAR ' (A B C))**
que vale A, y
> **(CDR ' (A B C))**
que vale (B C), no son nombres adecuados cuando se pretende obtener el primer elemento de una lista y el resto de la lista (la lista formada al quitar el primer elemento). Por eso:

> **(DE PRIMERO (LIS) (CAR LIS))**
> **(DE RESTO (LIS) (CDR LIS))**

serán las funciones que use cuando quiera hacer referencia al primer elemento de una lista y a su resto. Por ejemplo:

> **(PRIMERO ' (A B C))**
vale A, y
> **(RESTO ' (A B C))**
Vale (B C). Se trata de conseguir que los nombres de las funciones y de las variables indiquen lo que hacen o son. Ahora tu puedes cambiar los nombres de algunas funciones LISP si los que tienen no te gustan. Por ejemplo:

> **(DE SUMA (X Y) (+X Y))**
> **(DE RESTA (X Y) (SUB X Y))**
> **(DE MULTIPLICA (X Y) (* X Y))**
> **(DE DIVIDE (X Y) (/ X Y))**

De esta manera, (2*3)+15—(6/2) ahora se puede escribir como

> **(SUMA (MULTIPLICA 2 3)**
> **(RESTA 15**
> **(DIVIDE 6 2**
+**))**

Antes de empezar a escribir programas más grandes debemos conocer algunas otras funciones. La primera es COND, que es algo semejante a la instrucción IF condicion THEN instruccion ELSE instruccion. La forma general de la función COND es:

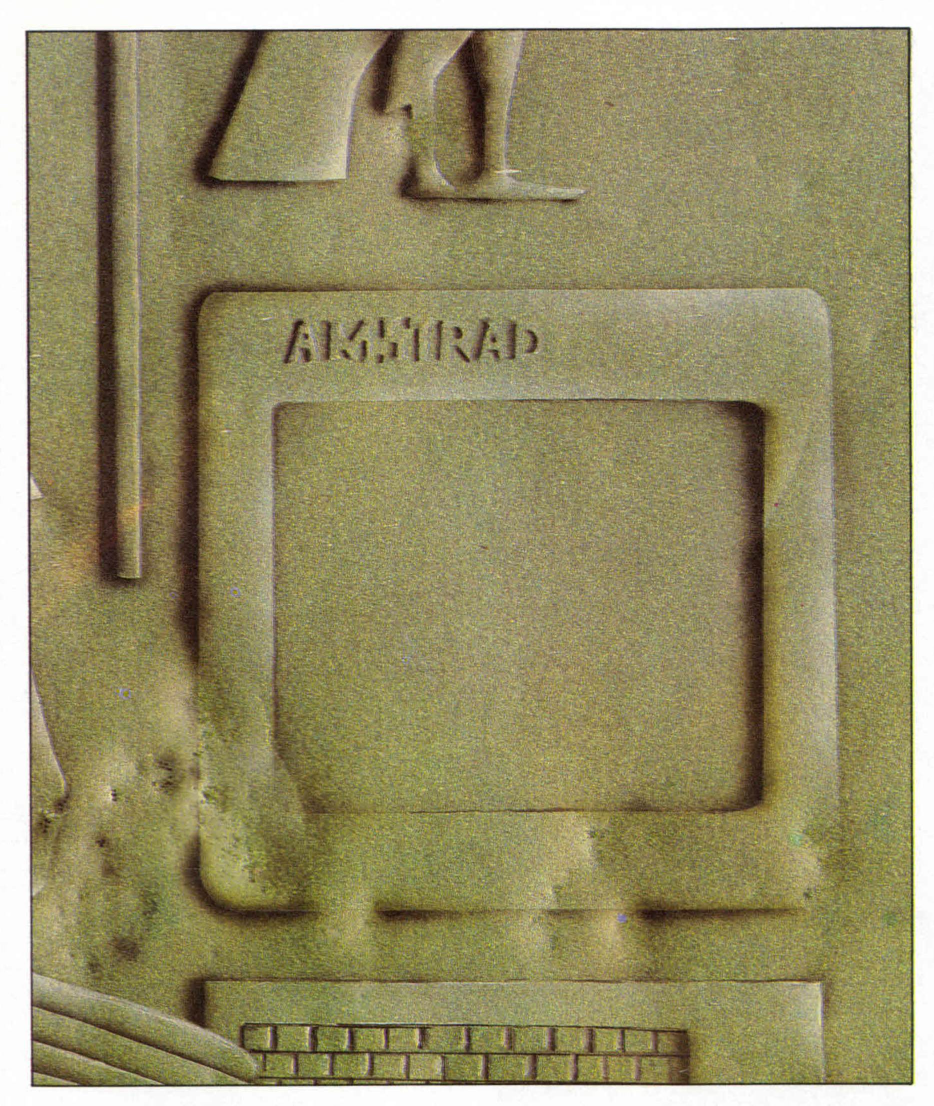
(COND (condición1 instrucción1)
(condición2 instrucción2)
...
(condiciónk instrucciónk))

donde condición1, instrucción1, ..., condiciónk, instrucciónk deben ser sustituidos por llamadas a funciones. En Basic se escribiría así:

IF condición1 THEN instrucción1
ELSE IF condición2 THEN instrucción2
...
ELSE IF condiciónk THEN instrucciónk

Si todas las condiciones valen NIL entonces el geniecillo te avisa de que han fallado todas las condiciones y el valor de la función COND es NIL. Si el valor de alguna de las condiciones no es NIL, entonces el valor de COND es el de la correspondiente instrucción. Veamos un ejemplo que nos aclara todo esto:

> **(DE SEGUNDO (LIS)**
> **(COND ((NULL LIS) ' (LA LISTA ES VACIA))**
> **((NULL (RESTO LIS) ' (LA LISTA SOLO TIENE UN ELEMENTO))**
> **(T (PRIMERO (RESTO LIS)))**
+**))**



Inteligencia ARTIFICIAL

Pidámosle al genio que nos diga cuál es el segundo elemento de algunas listas.

> **(SEGUNDO ())**

nos responde (LA LISTA ES VACIA), ya que ahora LIS vale () luego (NULL LIS) vale T, que es distinto de NIL, y por tanto el valor de COND es el de evaluar ' (LA LISTA ES VACIA).

> **(SEGUNDO ' (A))**

es (LA LISTA SOLO TIENE UN ELEMENTO). Ahora LIS vale (A), luego (NULL LIS) vale NIL y se pasa a evaluar la segunda condición. Como (RESTO LIS) es () se verifica (NULL (RESTO LIS)), que vale T y, por tanto, el valor de COND es el de evaluar ' (LA LISTA SOLO TIENE UN ELEMENTO).

> **(SEGUNDO ' (A B))**

vale B, ya que LIS vale (A B), (NULL LIS) es NIL y (RESTO LIS) es (B), luego (NULL (RESTO LIS)) es NIL. Como T vale T, el valor de COND es el de la tercera instrucción, o sea, el resultado de evaluar (PRIMERO (RESTO LIS)), que es lo mismo que (PRIMERO ' (B)), es decir, B.

El truco de poner T como última condición sirve para que siempre se verifique alguna condición. Se suele poner casi siempre, pero no es necesario.

De igual forma que en Basic, en LISP puedes utilizar las funciones booleanas AND, OR y NOT. AND y OR admiten cualquier número de parámetros, mientras que NOT sólo tiene uno. AND vale T si ninguno de sus argumentos vale NIL y vale NIL si alguno de sus argumentos vale NIL.

+ **(AND (3 + = 1) (1 < + 23))**

vale T.

+ **(AND (NULL ()) (CAR ' (NILL)))**

vale NIL, pues (CAR ' (NILL)) es NIL.

La función OR toma el valor del primer argumento cuyo valor no sea NIL y vale NIL si todos sus argumentos lo valen.

+ **(OR (NULL ' (A)) (CAR ' (B C)))**

vale B, pues (NULL ' (A)) es NIL, pero (CAR ' (B C) es B, que es distinto de NIL.

La función NOT vale T, si el valor de su argumento es NIL, y vale NIL cuando el valor de su argumento no sea NIL.

Si has estado jugando con el genio de MINILISP habrás visto que no es necesario que le digas que escriba un valor para que lo haga. Sin embargo, todos los mensajes que aparecen en la pantalla pueden ser molestos en ciertas ocasiones (cuando quieras que tu programa tenga una «salida» agradable, por ejemplo). En ese caso, si escribes (PREVAL NIL) suprimirás todos los mensajes que el genio te muestra en pantalla (excepto su saludo de «listo»). Cuando quieras que los mensajes aparezcan de nuevo escribe (PREVAL T), por ejemplo. Si has quitado los mensajes, el genio no escribirá nada si tú no se lo dices. Para eso están las funciones de entrada y salida de datos por pantalla: PRINT, PRINT1, TERPRI y READ.

La función PRINT escribe el valor de su único parámetro y no salta de línea. Como toda función LISP, debe tener un valor que es el de su argumento. Si escribes:

+ **(PREVAL NIL)**
+ **(PRINT ' ESCRIBO)**

aparecerá en pantalla.

ESCRIBO +

pues no ha saltado de línea. La función PRINT1 es igual que PRINT, pero con salto de línea. Luego:

+ **(PRINT1 ' ESCRIBO)**
ESCRIBO
+

La función TERPRI no tiene parámetros, vale NIL y su efecto es producir un salto de línea.

Para leer una S-expresión se utiliza la función READ, que tampoco tiene parámetros y cuyo valor es la S-expresión que ha leído.

+ **(PRINT1 (READ))**
+

y se queda esperando que le des una S-expresión. Dale cualquiera, por ejemplo:

+ **(ESCRIBE)**
(ESCRIBE)
+

Otra función de entrada/salida es CLS. La llamada:

+ **(CLS)**

borra la pantalla.

Antes hable de los programas que podían ser generados y evaluados desde otro programa LISP. Construir programas ya sabemos

+ **(LIST ' SUMA 2 2)**

vale (SUMA 2 2), pero no 4. Para evaluar esta lista se utiliza una nueva función, llamada EVAL.

> **(EVAL (LIST ' SUMA 2 2)**

si vale 4.

Explicaré ahora el ejemplo que abre este artículo.

```
+ (DE EQUAL (SEX1 SEX2)
+      (COND ((ATOM SEX1) (EQ
SEX1 SEX2))
+                ((EQUAL (CAR SEX1)
(CAR SEX2))
+                 (EQUAL (CDR SEX1)
(CDR SEX2)))
+                 (T NIL)
+ ) )
```

EQUAL es una función con dos parámetros, dos S-expresiones cualesquiera. Su valor es T si las dos S-expresiones SEX1 y SEX2 son igua-

les y NIL en otro caso. Si el primer SEXo es un átomo entonces el valor de EQUAL es el de (EQ SEX1 SEX2). Recordemos que EQ era cierto, T, si sus dos argumentos eran dos átomos literales iguales y NIL en otro caso. Por tanto, si SEX2 no es un átomo literal o, siéndolo, es distinto de SEX1 no es un átomo, pero si lo es SEX2 entonces el valor de EQUAL es NIL, como debe ser. Si ninguno de los SEXos es un átomo, entonces ambos deben ser pares o listas. Ninguno puede ser la lista vacía (pues (), o NIL, también es un átomo) luego podemos tomar sus CAR y sus CDR. Para que dos listas sean iguales elemento a elemento debe ocurrir que tengan el primer elemento igual y los restos de sus listas también sean iguales. Si sus primeros elementos no son iguales entonces las listas ya no serán iguales.

El esquema de la función EQUAL representa una forma usual de programar en LISP:

Para calcular el valor de una función aplicada a una lista hacer: Mientras la lista no sea vacía hacer:

Aplicar alguna operación sobre el primer elemento de la lista; aplicar la función al resto de la lista.

La función EQUAL es una función del sistema y que no es necesario que la programes. Sin embargo, si quieres ver como funciona puedes definirla con otro nombre, por ejemplo, IGUALES:

+ **(DE IGUALES (SEX1 SEX2)...**

Ahora escribe

+ **(TRACE ' (IGUALES))**

y llama a IGUALES con algunos argumentos algo complicados.

+ **(IGUALES ' (A (B (C.D) E) () F)**
+ **' (A (B (C.D) E) ()))**

Irán apareciendo en pantalla los sucesivos argumentos de la función IGUALES y sus respectivos valores. Cuando ya no quieras ver la **«traza»** (TRACE) de la función, puedes escribir:

+ **(UNTRACE ' (IGUALES))**

La función IGUALES, tal como está definida, no funciona para números, es decir:

+ **(IGUALES 5 5)**

vale NIL.

Edición de programas

Existen en MINILISP algunas facilidades para una mejor escritura de programas:

i) ' SEX es una abreviatura de (QUOTE SEX).

El símbolo ' no puede formar parte de un identificador pues PEPE'S es interpretado por MINILISP como:

PEPE (QUOTE S)

ii) % hace que MICROLISP ignore el resto de la línea. Sirve para escribir comentarios.

iii) / cierra todos los paréntesis que quedan por cerrar. Se produce un error si ya estaban todos cerrados.

La edición de programas LISP es una tarea costosa, de manera que MINILISP no lleva incorporado editor. Por ello te recomendamos que cuando sepas que vas a escribir lo hagas con un editor de textos y luego le pases el fichero donde está tu programa a MINILISP [luego explicaremos como puede hacerse esto]. Desgraciadamente los programas no suelen funcionar a la primera, así es que tendrás que corregirlos y suele ser más cómodo hacer esto ayudado por MINILISP. Para ello te recomendamos que sigas los siguientes pasos:

1. Borra la definición incorrecta con la llamada:

(REMPROP ' < nombre + 'EXPR)

donde < nombre + es el nombre de la función errónea. Esta llamada devuelve la lambda-expresión que define a la función.

2. Escribe

(PUT ' < nombre + 'EXPR'

y a continuación copiar con las teclas del cursor la lambda-expresión que obtuvimos antes, corrigiendo la parte que esté mal. [Si la definición es muy larga y no cabe en 255 caracteres tendrás que pulsar alguna vez la tecla [ENTER], pero ten cuidado de no partir algún identificador ya que, entonces, MINILISP lo consideraría como dos átomos].

3. Te puedes evitar escribir los paréntesis finales si escribes el **símbolo** / .

4. Pulsa [ENTER].

Funciones de comunicación externa

MINILISP puede mantener abiertos un fichero de entrada, uno de salida y una impresora. Para abrir y cerrar estas vías de comunicación se utilizan las funciones OPEN y CLOSE. Las dos funciones son de tipo e.s.

OPEN tiene dos argumentos. El primero debe tener como valor un átomo, que junto con la extensión .LIS se considera como nombre del fichero. El segundo argumento es el modo de operación: INPUT, OUTPUT, DEFS, PROPS o PRINTER.

Si el modo es INPUT el fichero se abre como fichero de entrada [desde disco o cinta]. En este modo, cada vez que MINILISP intente leer lo hará del fichero establecido. Cuando se alcanza el final de fichero, automáticamente MINILISP vuelve a establecer como fichero de entrada el teclado de la computadora [este hecho lo advierte con un pitido]. Para detectar el final del fichero se puede utilizar la función EOF, que no tiene argumentos. Su valor es T, si se ha alcanzado el final del fichero, y NIL en otro caso.

Si el modo es OUTPUT, se abre el fichero como fichero de salida y todo lo que MINILISP envíe a pantalla lo mandará también al fichero.

Con el modo DEFS se abre un fichero de salida en el que se guardan las definiciones de las funciones de usuario que MINILISP conozca hasta ese momento. Cada definición se guarda escrita en la forma (DE f (x1...xk) c) por lo que se genera un fichero que puede ser **«comprendido»** por MINILISP.

Si el fichero se abre en modo PROPS, es considerado como fichero de salida y en el que se escriben las propiedades de los identificadores que MINILISP conozca hasta el momento. Estas propiedades se guardan en la forma (PUT 'id 'prop 'val) con lo que se genera un fichero que puede ser leído por MINILISP. De esta forma se puede preservar el contexto actual y reanudar posteriormente la sesión desde el punto en el que dejó. Para ello bastará hacer (OPEN ' < nombre + 'INPUT) donde < nombre + es el nombre del fichero [sin la extensión, ya que se asume .LIS].

El modo PRINTER sirve para mandar la salida por pantalla también a la impresora. Además, si se abre algún fichero en los modos DEFS o PROPS, aquello que se escriba también en estos ficheros se manda a la impresora.

La función CLOSE cierra el fichero que esté abierto con el modo especificado por su parámetro. El valor del argumento debe ser INPUT, OUTPUT o PRINTER. Tiene como valor el de su argumento.

La función CAT, que no tiene parámetros, sirve para obtener un catálogo del disco o la cinta. Si se está utilizando cinta debe pulsarse la tecla [ESC] para terminar el catálogo. Si se usa disco esto no es necesario.

NOTA PARA LA CARGA DE LA CINTA

— El LISP de la cara A de la cinta es para usuarios del 664 y 6128. No es 72 utilizable directamente sino que le obligará a efectuar una copia en Disco. Siga las instrucciones que se le vayan dando.

— La cara 2 es para usuarios 464.

— No obstante al principio de cada cara puede usted ganar un ordenador. carguelas sea cual sea su modelo.

— Minilisp tarda en inicializarse una vez cargado, espere a que salga el promp.

— En ambas caras se encuentra información sobre el gran concurso AMSTRAD SEMANAL-OFITES INFORMATICA.

Manual de Minilisp

Ofrecemos aquí un manual de referencia que sirva sobre todo, cuando ya domines el lenguaje, para resolver todas aquellas pequeñas dudas que siempre surgen.

R. G. Bernal

1 MINILISP: Es el encargado de cargar el resto del programa. También inicializa el tamaño de la tabla de símbolos.

2. LISP. BAS: Es el intérprete de LISP [en Basic]. Permanece constantemente en memoria [no así el anterior, que se borra al cargarse éste].

3. LISP. DAT: Este es un fichero ASCII que contiene los datos necesarios para rellenar el contexto inicial.

4. ERRLISP. BAS: Contiene la parte de gestión de errores y mensajes del sistema. Se carga tras borrar de la memoria la rutina de inicialización de las estructuras.

2.° Tipos de datos primitivos

Todas las consecuencias admisibles por MINILISP son expresiones simbólicas [S-expresiones]. Entre éstas podemos distinguir varios tipos y sus reconocedores:

S-expresiones:
- Atomos: ATOM
  - Literales: LITATOM
  - Números: NUMBERP
- Pares: PAIRP

Sintaxis de las S-expresiones

Para definir la sintaxis de un lenguaje de programación se suelen utilizar algunos formalismos como las reglas de Backus-Naur o los diagramas sintácticos. Aquí emplearemos el primero de ellos.

Las reglas en forma de Backus-Naur [reglas FBN] constan de tres partes: un lado izquierdo, que es lo que se quiere definir: el símbolo ::=, y un lado derecho, que es la definición. Los objetos que se quiere definir se suelen encerrar entre los símbolos < y >, por ejemplo < Número2 >. La yuxtaposición de objetos **[escribir uno al lado de otro]** indica que a continuación de un objeto del primer tipo se escribe uno del segundo. Por ejemplo:

< Número2 > ::= < Dígito > < Dígito >

indica que un objeto de tipo < Número2 > se representa por un objeto de tipo < Dígito > seguido por otro del mismo tipo. La barra vertical ! actúa como disjuntor [0 lógico]. Por ejemplo:

< Dígito > ::= 0 ! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 ! 9

quiere decir que un objeto de tipo < Dígito > es o bien 0, o bien 1, o bien..., o bien 9. Entonces, ejemplos de objetos de tipo < Número2 > serían 55, 13, 17, 80, 01, 00 y 99.

Otros símbolos que se usan en las reglas FBN son [,] y {, }. Los nombres de objetos se encierran entre corchetes para indicar que es opcional escribirlos o no. Por ejemplo:

< Número2—con—signo > ::= [< Signo >] < Número2 >

quiere decir que un objeto de tipo < Número2—con—signo > es un objeto de tipo < Número2 > que puede ir precedido o no de un signo. Si definimos:

< Signo > ::= + I—

entonces ejemplos de objetos de tipo < Número2—con—signo + son —34, +13, —00, +00, 00, 99 y 17.

Las llaves { y } indican que el objeto al que encierran se puede repetir tantas veces como se quiera. Por ejemplo:

< Número—sin—signo + ::= < Dígito + { < Dígito + }

quiere decir que un objeto de tipo < Número2—sin—signo + es una secuencia de objetos de tipo < Dígito + de cualquier longitud [debe tener al menos un dígito]. Por ejemplo, 12345567890, 1111111111111111111111111 1111111111, etc. Aunque conceptualmente el tamaño del objeto no esté limitado, físicamente si que lo está (número de bytes de un computador, número de folios que puedes escribir, etc.).

Si has entendido todo esto no tendrás problemas en comprender las reglas FBN que definen la sintaxis de las S-expresiones y que damos a continuación. Sin embargo, si no las entiendes no te preocupes, pues más adelante se dan ejemplos.

< S-expresión + ::= < Atomo + I < Par + I < Lista +
< Atomo + ::= < Literal + I < Numeral +
< Literal + ::= < Marca + < Atomo +
< Numeral + ::= [< Signo +] < Dígito + { < Dígito + }
< Par + ::= (< S-expresión + . < S-expresión +)
< Lista + ::= () I (< Lista—de— objetos +
< Lista—de—objetos + ::= < S-expresión + { < Lista—de—objetos + }
< Marca + ::= < Letra + I < Símbolo +
< Dígito + ::= 0 I 1 I 2 I 3 I 4 I 5 I 6 I 7 I 8 I 9
< Signo + ::= + I—
< Letra + ::= A I B I C I D I E I F I G I H I I I J I K I L I M I N I O I P I Q I R I S I T I U I V I W I X I Y I Z I a I b I c I d I e I f I g I h I i I j I k I l I m I n I o I p I q I r I s I t I u I v I x I y I z I
< Símbolo + ::= ! I '' I # I $ I & I * I + I , I / I : I ; I < I = I + I ? I ¶ I [I] I ^ I — I { I I I } I #

Obsérvese que como símbolos no se admiten:
- Operador unario: —
- Contracción de QUOTE:
- Terminador de paréntesis:
- Comentario: %
- Punto del par: .

Veamos algunos ejemplos:

Literales

suma + suma—+ ES—suma? +suma *no—Suma dolar$

Numerales

00000000 1523 —153 -00 0000001 32767 —32768

[En MINILISP los ceros a la izquierda no sirven para nada. Hay que notar también que números mayores que 32767 o menores que —32768 darán un error 6 de desbordamiento numérico.]

Atomos

Todos los ejemplos anteriores lo son también de átomos, pues los átomos o son literales o son numerales.

Pares

(suma. +) (UNO.1) (PAR. (uno.1)) ((par. de). (dos.pares))
(JOSE. (quiere.(a.(ANA.NIL))))

Listas

(JOSE quiere a ANA) ((LISTA.DE) (DOS.0) (TRES.PARES))

(LISTA (DE (LISTAS (DE (MAS) LISTAS))))

S-expresiones

Todos los ejemplos lo son también de S-expresiones, por la primera regla.

3.° Constantes predefinidas

En el contexto inicial de MINILISP se han introducido un reducido conjunto de constantes necesarias para el uso del intérprete. Podemos dividirlas en dos tipos, según que tengan o no atributo VALUE:

1. Constantes con valor:
NIL: valor NIL
T: valor T
PIZQ: valor (
PDER: valor)

2. Constantes sin valor:
VALUE, LAMBDA, FSUBR, SUBR, EXPR, FEXPR y MACRO.

4.° Lista de objetos y listas de propiedades

MINILISP mantiene una lista de los identificadores conocidos hasta el momento de una tabla hash. En cualquier momento es posible ver el contenido de esta tabla mediante la llamada:

(OBLIST)

Sin embargo, por razones de economía de memoria y de velocidad de proceso, la lista de objetos no se mantiene en una lista LISP por lo que OBLIST tiene valor NIL y como efecto colateral la impresión de la tabla de identificadores.

MINILISP inserta un identificador en la tabla de símbolos sólo y cuando éste no se encuentra ya en la tabla. Por tanto, el predicado de igualdad de átomos, EQ, se implementa mediante la comprobación de igualdad de punteros. Por este motivo, EQ no es válido para verificar ni la igualdad de pares ni la de números. Para pares debe utilizarse la función EQUAL. La igualdad entre números se comprueba con el predicado =.

Funciones que modifican la lista de objetos:

(REMOVE id) id debe evaluar a un átomo. Tiene como efecto suprimir el identificador id de la lista de objetos. Su valor es NIL.

(REMOVE 'PRIMERO) eliminaría el átomo PRIMERO de la lista de objetos y hace inaccesibles todas sus propiedades.

(COMPRESS lis) lis debe tener como valor una lista de átomos, debiendo ser el primero de estos un átomo literal. Comprime la lista para formar un identificador literal. El átomo resultante se coloca en la lista de objetos y se devuelve como valor de la función.

(COMPRESS '(VAR 1 4)) devuelve como valor VAR14 a la vez que inserta este átomo en la lista de objetos (si no estaba ya).

(EXPLODE id) id debe evaluar a un átomo. Tiene el efecto contrario a COMPRESS: genera la lista de los caracteres que componen id, que se da como valor de la función.

Los caracteres que no son dígitos se colocan en la lista de objetos.

(EXPLODE 'PALABRA) vale (P A L A B R A) y coloca los átomos P, A, L, B y R en la lista de objetos (si aún no estaban).

(EXPLODE 'VAR14) vale (V A R 1 4) e introduce los átomos V, A y R en la lista de objetos (si no estaban).

(EXPLODE 'A-Z) vale (A I-Z) pues el símbolo - no está permitido como átomo.

Listas de propiedades:

MINILISP mantiene una lista de propiedades (atributos) para cada identificador presente en la lista de objetos.

Existen tres funciones en MINILISP encargadas de crear, consultar y modificar listas de propiedades. Todas ellas evalúan sus argumentos:

(PUT id prop val) Coloca el resultado de evaluar val como valor de la propiedad prop del identificador id. Si ya existía la propiedad, da error y no modifica el valor de prop.

(PUT 'PRIMERO 'EXP' (LAMBDA (LIS) (CAR LIS))) coloca la lambda-expresión (LAMBDA (LIS) (CAR LIS) como valor del atributo EXPR en la lista de propiedades del identificador PRIMERO. El valor que devuelve es PRIMERO.

(GET prop id) Devuelve el valor de la propiedad prop en la lista de propiedades de id [o NIL si no aparece en la lista de propiedades de id].

(GET 'EXPR 'PRIMERO) vale (LAMBDA (LIS) (CAR LIS) si se hizo la instrucción anterior y NIL si no existe la propiedad EXPR de PRIMERO.

(REMPROP id prop) Elimina prop y su valor val de la lista de propiedades de id.

Devuelve val [o id y un mensaje si prop no existía como propiedad id].

'REMPROP 'PRIMERO 'EXPR) elimina la propiedad EXPR de la lista de propiedades de PRIMERO, y tiene como valor la S-expresión (LAMBDA (LIS) (CAR LIS)). Si después de esta llamada se hace (GET 'EXPR 'PRIMERO) se devuelve el valor NIL pues ya no existe la propiedad 'EXPR.

5.° Definición de funciones y macros

Funciones:

De los cuatro tipos posibles de funciones en LISP [según que se evalúen o no los argumentos y que el número de éstos sea fijo o variable] sólo dos son directamente definibles en MINILISP: eval-spread [evalúa y número de argumentos fijo] y noevalnospread [no evalúa y número de argumentos variable]. Ninguno de estos «definidores» evalúa sus argumentos:

(DE f (x1... xn) c) Define la función f como de tipo e.s. con variables formales x1, ..., xn y cuerpo c. Tiene como valor f y su efecto es poner la lambda-expresión (LAMBDA (x1 ... xn) c) como valor del atributo EXPR de f.

(DF f (1) c) Define la función f como de tipo n.n. con cuerpo c y lista de argumentos 1. Su efecto es poner la lambda-expresión (LAMBDA (1) c) como valor de la propiedad FEXPR de f. Tiene como valor f.

Macros:

Es este un tipo especial de definición. En MINILISP se puede utilizar mediante la función DM con la siguientes sintaxis:

(DM f (1) c) No evalúa sus argumentos. Tiene valor f y como efecto pone la lambda-expresión (LAMBDA (1) c) como valor del atributo MACRO de f.

Las llamadas a una macro son de la forma:

(f e1... en) (n cualquiera)

y se evalúan en dos fases:

i) Se evalúa c en un contexto ampliado con la ligadura de 1 a la forma (f e1... en).

ii) Se evalúa la forma obtenida en i) siendo el resultado el valor de la llamada.

Un ejemplo de macro es la función LET:

```
(DM LET (LA) (CONS (LIST 'LAMDA
     VARIABLES (PARES LA))
            (EXPCUALIF LA))
     (EXPRESIONES (PARES LA))))
```

siendo

```
(DE PARES (U) (RESTO (RESTO U)))
(DE VARIABLES (LISPAR)
(COND ((NULL LISPAR( ())
          (T (CONS (VAR (PRIME-
                RO LISPAR))
                     (VARIABLES
                        (RESTO
                       LISPAR))))
))
(DE EXPRESIONES (LISPAR)
(COND ((NULL LISPAR) ())
       (T (CONS (EXP (PRIMERO
       LISPAR))
              (EXPRESIONES
              (RESTO
              LISPAR))))
))
(DE EXPCUALIF (U) (SEGUNDO U))
(DE VAR (PAR) (PRIMERO PAR))
(DE EXP (PAR) (SEGUNDO PAR))
(DE PRIMERO (LIS) (CAR (LIS)
(DE SEGUNDO (LIS) (PRIMERO
(RESTO LIS)))
(DE RESTO (LIS) (CDR LIS))
```

Supongamos que queremos evaluar la expresión:

```
(LET (LIST U V U)
     (U (CAR '(A B C)))
     (V (CDR '(D E F))))
```

En el primer paso se da como valor del parámetro LA de la función LET el valor

(LET (LIST U V U) (U (CAR '(A B C))) (V CDR '(D E F))))

y se evalúa de forma usual el cuerpo de la función LET. El resultado de esta evaluación es:

(LAMBDA (U V) (LIST U V U) (CAR '(A B C)) (CDR '(D E F)))

En el segundo paso se evalúa esta expresión dando como valor

(A (E F) A)

que pasa a ser el valor de la función LET.

6.° Construcciones iterativas

Aunque la programación imperativa no es necesaria en un lenguaje funcional como LISP su uso, no obstante, está justificado en un pequeño computador como en el que se ejecuta MINILISP, tanto por el ahorro de memoria [que no es excesiva y no se puede «derrochar»] como de tiempo [no prolongar mucho programas ya que de por sí lentos]; incluso hay problemas técnicos [como el del tamaño de la pila que soporta la recursión].

En la base de estas construcciones se hallan las funciones de asignación:

(SET e1 e 2) De tipo e.s., la evaluación de e1 debe ser una variable formal o declarada. Liga al valor de e1 el de e2 y devuelve como valor el de e2.

(SETQ id e) Es de tipo n.n. Tiene el mismo efecto y valor que (SET 'id e).

Los **«constructores»** de instrucciones iterativas son dos, ambos de tipo n.n.:

PROGN e1... ek)

Permite la llamada consecutiva a varias funciones. Tiene como valor el de ek [e1, ..., e(k-1) se evalúan por su «efecto»].

(PROG (x1...xn) e1...ek)

x1, ..., xn son las variables locales de la forma PROG (sólo se pueden utilizar dentro del cuerpo de la instrucción PROG). Las formas e1, ..., ek constituyen el cuerpo de la forma PROG.

La lista de variables puede ser (), pero debe existir. Las variables de una forma PROG se inicializan todas a NIL.

Las formas que aparecen en el cuerpo de una forma PROG pueden ser: condicionales, formas aplicativas, formas RETURN, formas GO, etiquetas y asignaciones.

Las formas condicionales no necesitan tener un lado izquierdo cierto [en una forma condicional fuera de un PROG, si todos los lados izquierdos se evalúan a NIL se genera un mensaje de aviso]. En este caso toma el valor NIL y se cede el control a la siguiente instrucción.

(RETURN e) Es de tipo e.s. Termina la evaluación del PROG, devolviendo como valor el de e.

(GO id) Es de tipo n.n. Cede el control a la instrucción siguiente a la etiqueta id [para MINILISP, una etiqueta es cualquier átomo literal]. Si id no es una etiqueta o no existe dentro de la forma PROG se genera un error.

Los argumentos de PROG son evaluados uno tras otro, a partir del segundo [el primero es la lista de variables], con las siguientes excepciones:

a) Un argumento atómico no se evalúa: se considera como una etiqueta.

b) La evaluación de la función GO hace que LISP continúe la evaluación de PROG tras la etiqueta que es el argumento de GO.

c) Si la función RETURN se evalúa, entonces acaba la evaluación de PROG, y el valor de la función PROG es el valor del argumento de RETURN.

d) Si el último argumento de PROG ha sido evaluado y no era ni una forma GO ni una forma RETURN entonces el valor de PROG es NIL.

e) Si un argumento de PROG es un COND y ninguno de sus lados izquierdos vale T entonces la evaluación continúa con el siguiente argumento de PROG.

7.° Aritmética y lógica

Aritmética

Tanto los operadores aritméticos como los relacionales entre números son de tipo e.s. [Los números se evalúan a sí mismos] y tienen dos argumentos. Son los siguientes:

(+ e1 e2) e1+e2
(SUB e1 e2) e1—e2
(* e1 e2) e1*e2
(DIV e1 e2) e1/e2 [división entera]
(ê1 e2) $e1^{e2}$
(MAX e1 e2) el máximo entre e1 y e2

Los operadores relacionales son:

= + < += <=

Lógica

Las funciones lógicas existentes en MINILISP son:

(NOT e). Es de tipo e.s. Coincide con la función NULL: vale T, si (EQ e NIL), y NIL, en otro caso.

(AND e1... ek). Es de tipo n.n. Para calcular su valor se evalúan las ei en orden, desde la izquierda hacia la derecha. Si alguna da NIL se devuelve NIL y no se evalúan las demás; si no, se devuelve T.

(OR e1... ek). Es de tipo n.n. Para calcular su valor se evalúan en orden las ei. Devuelve el valor del primer ei que no sea NIL, o NIL si todos los ei valen NIL.

8.° Funciones MAP

Las mejoras introducidas en esta nueva versión de MINILISP permiten que se utilicen argumentos funcionales, de ahí que sea posible la definición de funciones map. Por razones de espacio el intérprete de MINILISP no lleva incorporada ninguna.

Un ejemplo de función map es el siguiente:

```
        (DE MAP (LIS FUN)
      (COND ((NULL LIS) NIL)
(T (PROGN (FUN LIS) (MAP (RESTO
```

LIS) FUN))
))

FUN es el argumento funcional, es decir, el valor de FUN es o bien el nombre de una función o una lambda-expresión. Entonces (FUN LIS) quiere decir que el valor de FUN se aplica al valor de LIS [se espera que LIS sea una lista]. Otros ejemplos de funciones MAP pueden verse en el fichero de ejemplo SURTIDO.LIS.

9.° Funciones de edición

Existen en MINILISP una serie de facilidades para la mejor escritura de programas:

i) 'SEX es una abreviatura de (QUOTE SEX).

El símbolo ' no puede formar parte de un identificador pues PEPE'S es interpretado por MINILISP como:

PEPE (QUOTE S)

ii) % hace que MICROLISP ignore el resto de la línea. Sirve para escribir comentarios.

iii) / cierra todos los paréntesis que quedan por cerrar. Se produce un error si ya estaban todos cerrados.

La edición de programas LISP es una tarea costosa, de manera que MINILISP no lleva incorporado editor. Por ello te recomendamos que cuando sepas que vas a escribir lo hagas con un editor de textos y luego le pases el fichero donde está tu programa a MINILISP. Desgraciadamente los programas no suelen funcionar a la primera, así es que tendrás que corregirlos y suele ser más cómodo corregirlos ayudado por MINILISP. Para ello te recomendamos que sigas los siguientes pasos:

1. Borra la definición incorrecta con la llamada:

(REMPROP ' < nombre + 'EXPR)

donde < nombre + es el nombre de la función errónea. Esta llamada devuelve la lambda-expresión que define a la función.

2. Escribe (PUT '< nombre + 'EXPR' y a continuación copiar con las teclas del cursor la lambda-expresión que obtuvimos antes, corrigiendo la parte que esté mal. [Si la definición es muy larga y no cabe en 255 caracteres tendrás que pulsar alguna vez la tecla [ENTER], pero ten cuidado de no partir algún identificador ya que, entonces, MINILISP lo consideraría como dos átomos].

3. Te puedes evitar escribir los paréntesis finales si escribes el símbolo /.

4. Pulsa ENTER.

10.° Entrada y salida

(READ). No tiene argumentos. Su valor es la primera S-expresión que lee del dispositivo actual de entrada.

(PRINT e). Es de tipo e.s. Escribe, con salto de línea, el valor de e, que también es el valor de la función.

(PRINT1 e). Igual que PRINT, pero sin salto de línea.

(TERPRI). No tiene argumentos. Tiene como valor NIL y como efecto produce un salto de línea en los dispositivos de salida establecidos.

(SPACE e). Es de tipo e.s. El valor de e debe ser un número que es el número de caracteres blancos que deja. [No tiene salto de línea]. El valor de la función es el mismo que el de e.

Funciones de comunicación externa

MINILISP puede mantener abiertos un fichero de entrada, uno de salida y una impresora. Para abrir y cerrar estas vías de comunicación se utilizan las funciones OPEN y CLOSE. Las dos funciones son de tipo e.s.

OPEN tiene dos argumentos. El primero debe tener como valor un átomo, que junto con la extensión .LIS se considera como nombre del fichero. El segundo argumento es el modo de operación: INPUT, OUTPUT, DEFS, PROPS o PRINTER.

Si el modo es INPUT el fichero se abre como fichero de entrada [desde disco o cinta]. En este modo, cada vez que MINILISP intente leer lo hará del fichero establecido. Cuando se alcanza el final de fichero, automáticamente MINILISP vuelve a establecer como fichero de entrada el teclado de la computadora [este hecho lo advierte con un pitido]. Para detectar el final del fichero se puede utilizar la función EOF, que no tiene argumentos. Su valor es T, si se ha alcanzado el final del fichero, y NIL en otro caso.

Si el modo es OUTPUT, se abre el fichero como fichero de salida y todo lo que MINILISP envíe a pantalla lo mandará también al fichero.

Con el modo DEFS se abre un fichero de salida en el que se guardan las definiciones de las funciones de usuario que MINILISP conozca hasta ese momento. Cada definición se guarda MINILISP conozca hasta ese momento. Cada definición se guarda en la forma (DE f (x1... xk) c) por lo que se genera un fichero que puede ser «comprendido» por MINILISP.

Si el fichero se abre en modo PROPS, es considerado como fichero de salida y en él se escriben las propiedades de los identificadores que MINILISP conozca hasta el momento. Estas propiedades se guardan en la forma (PUT 'id 'prop 'val) con lo que se genera un fichero que puede ser leído por MINILISP. De esta forma se puede preservar el contexto actual y reanudar posteriormente la sesión desde el punto en el que se dejó. Para ello bastará hacer (OPEN ' < nombre + 'INPUT) donde < nombre + es el nombre del fichero [sin la extensión, ya que se asume .LIS].

El modo PRINTER sirve para mandar la salida por pantalla también a la impresora. Además, si se abre algún fichero en los modos DEFS o PROPS, aquello que se escriba también en estos ficheros se manda a la impresora.

La función CLOSE cierra el fichero que esté abierto con el modo especificado por su parámetro. El valor del argumento debe ser INPUT, OUTPUT o PRINTER. Tiene como valor el de su argumento.

La función CAT, que no tiene parámetros, sirve para obtener un catálogo del disco o la cinta. Si se está utilizando cinta debe pulsarse la tecla [ESC] para terminar el catálogo. Si se usa disco esto no es necesario.

11.° Errores

MINILISP informa de dos tipos de errores: los producidos por él y los generados por Basic. De estos últimos sólo indica el número de error [estos números los puedes encontrar en el manual del ordenador]. Los errores producidos por Basic suelen ser errores numéricos [6, 11] o de espacio para cadenas lleno [14] si has utilizado identificadores excesivamente largos [se ha calculado que cada identificador ocupará en media 8 caracteres], pero eventualmente es posible que aparezcan errores de otro tipo si se ha producido algún fallo anterior.

Los errores generados por fallos en los programas LISP están documentados con mensajes de error y avisos.

AMSTRAD

MINILISP DICCIONARIO RAPIDO DE FUNCIONES

(* e1 e2)
Tipo: e.s.
Valor: el producto de e1 y e2

(+ e1 e2)
Tipo: e.s.
Valor: la suma de e1 y e2

(< e1 e2)
Tipo: e.s.
Valor: T si e1 < e2; error, si alguno no es número; NIL e.o.c.

(< = e1 e2)
Tipo: e.s.
Valor: T si e1 < =e2; NIL (o error si alguno no es número) e.o.c.

(= e1 e2)
Tipo: e.s.
Valor: T si e1 y e2 si son números iguales; NIL (o error) e.o.c.

(+ e1 e2)
Tipo: e.s.
Valor: T si e1 + e2; error, si alguno no es número; NIL e.o.c.

(+ = e1 e2)
Tipo: e.s.
Valor: T si e1 + =e2; NIL (o error si alguno no es número) e.o.c.

(AND e1...ek)
Tipo: e.n.
Valor: NIL si alguna de las ei se evalúa a NIL; T e.o.c.

(APPLY f 1)
Tipo: e.s.
Valor: el de aplicar la lambda-expresión f a la lista de argumentos 1

(ASSOC atm 1a)
Tipo: e.s.
Valor: el par de la lista de asociación la que tiene como CAR atm. Da NIL atm no aparece o no es un átomo (usa EQ)

(ATOM a)
Tipo: e.s.
Valor: T si a es un átomo; NIL e.o.c.

(ATOMLEN e)
Tipo: e.s.
Valor: el número de caracteres del átomo al que evalúa e

(CAR e)
Tipo: e.s.
Valor: el primer elemento del par e (error si es un átomo) e.o.c.

(CAT)
Tipo: n.n.
Valor: NIL
Efecto: genera un catálogo de los ficheros accesibles

(CDR e)
Tipo: e.s.
Valor: el segundo elemento del par e (error si es un átomo)

(CLOSE m)
Tipo: e.s.
Valor: el de m
Efecto: **-f4-**cierra el dispositivo abierto como el modo m

(CLS)
Tipo: n.n.
Valor: NIL
Efecto: **-f4-**borra la pantalla

(COMPRESS e)
Tipo: e.s.
Valor: el átomo resultado de comprimir la lista e
Efecto: **-f4-**introduce el átomo en la lista de objetos

(COND (el a1)...(ek ak))
Tipo: e.n.
Valor: el de ai, siendo ei el menor e que se evalúa a T

(CONS e1 e2)
Tipo: e.s.
Valor: el par de CAR e1 y CDR e2

(DE f (x...xn) e)
Tipo: n.n.
Valor: f, sin evaluar
Efecto: **-f4-**semejante a (PUT 'f 'EXPR '(LAMBDA (x1...xk) e))

(DELETE sex lis)
Tipo: e.s.
Valor: la lista sin la primera aparición de sex

(DF f (1) e)
Tipo: n.n.
Valor: f, sin evaluar
Efecto: **-f4-**semejante a PUT 'f 'FEXPR '(LAMBDA) (1) e))

(DIV e1 e2)
Tipo: e.s.
Valor: el cociente (real) entre e1 y e2

(DM f (1) e)
Tipo: n.n.
Valor: f, sin evaluar
Efecto: **-f4-**semejante a (PUT 'f 'MACRO '(LAMBDA (1) E))

(DRAW x y)
Tipo: e.s.
Valor: el par formado por x e y evaluados
Efecto: **-f4-**traza una línea desde la posición actual del cursor de gráficos hasta el punto de coordenadas absolutas (x,y)

(DRAWR x y)
Tipo: e.s.
Valor: el par formado por x e y evaluados
Efecto: **-f4-**traza una línea desde la posición actual del cursor de gráficos hasta el punto de coordenadas absolutas (x,y)

(EOF)
Tipo: n.n.
Valor: T si se ha alcanzado el final del fichero de entrada; NIL

(EQ e1 e2)
Tipo: e.s.
Valor: T si e1 y e2 son el mismo átomo literal; NIL e.o.c.

(EQUAL e1 e2)
Tipo: e.s.
Valor: T si e1 y e2 son dos S-expresiones iguales; NIL e.o.c.

(EVAL e)
Tipo: e.s.
Valor: el de evaluar e

(EXPLODE e)
Tipo: e.s.
Valor: la lista de los caracteres que forman el átomo e
Efecto: **-f4-**introduce los caracteres no numéricos de e en la lista de objetos

(GET atr id)
Tipo: e.s.
Valor: el valor del atributo atr de id; NIL si atr no existe

(GO e)
Tipo: n.n.
Valor: NIL
Efecto: **-f4-**cede el control a la instrucción siguiente a la etiqueta e

(INFMEM)
Tipo: n.n.
Valor: NIL
Efecto: **-f4-**iforma del número de identificaciones y celdas libres

(LENGTH e)
Tipo: e.s.
Valor: la longitud de la lista resultado y celdas libres

(LIST e1...ek)
Tipo: e.n.
Valor: la lista formada por e1,..., ek

(LITATOM a)
Tipo: e.s.
Valor: T si a es un átomo literal; NIL e.o.c.

(MAX e1 e2)
Tipo: e.s.
Valor: el máximo entre e1 y e2

DICCIONARIO RAPIDO DE FUNCIONES MINILISP

(MEMBER sex lis)
Tipo: e.s.
Valor: T si S-expresión sex es un elemento [al nivel más externo] de la lista lis

(MEMQ atm lis)
Tipo: e.s.
Valor: el trozo de lis desde la aparición de atm hasta el final; NIL, si no aparece o no es un átomo

(MOD e1 e2)
Tipo: e.s.
Valor: el resto de la división entera de e1 y e2

(MOVE x y)
Tipo: e.s.
Valor: el par formado por los valores de x e y
Efecto: **-f4-**coloca el cursor de gráficos en las coordenadas absolutas (x,y)

(MOVER x y)
Tipo: e.s.
Valor: el par formado por los valores de x e y
Efecto: **-f4-**coloca el cursor de gráficos en las coordenadas relativas (x,y)

(INCONC lis1 lis2)
Tipo: e.s.
Valor: la lista formada concatenando lis2 al final de list1
Efecto: **-f4-**modifica el parámetro list1 pegándole físicamente lis2

(NOT e)
Tipo: e.s.
Valor: T si e se evalúa a NIL; NIL e.o.c.

(NULL e)
Tipo: e.s.
Valor: T si e es NIL; NIL e.o.c.

(NUMBERP n)
Tipo: e.s.
Valor: T si n es un átomo no literal; NIL e.o.c.

(OBLIST)
Tipo: n.n.
Valor: NIL
Efecto: **-f4-**escribe la lista de objetos en pantalla

(OPEN f m)
Tipo: e.s.
Valor: el de f
Efecto: **-f4-**abre un fichero de salida en el modo m

(OR e1...ek)
Tipo: e.n.
Valor: T si alguna de las ei se evalúa a T; NIL e.o.c.

(PAIRLIS lis1 list2)
Tipo: e.n.
Valor: la lista formada por los pares de elementos respectivos de lis1 y lis2. Da error si lis2 tiene menos elementos que lis1

(PAIRP p)
Tipo: e.s.
Valor: T si p es un par; NIL e.o.c.

(PLOT x y)
Tipo: e.s.
Valor: el para formado por los valores de x e y
Efecto: **-f4-**pinta un punto en las coordenadas absolutas (x,y)

(PLOTR x y)
Tipo: e.s.
Valor: el para formado por los valores de x e y
Efecto: **-f4-**pinta un punto en las coordenadas (x,y)

(PREVAL e)
Tipo: e.s.
Valor: el de e
Efecto: **-f4-**se desactivan los mensajes de EVAL si e es NIL; se activan e.o.c.

(PRINT e)
Tipo: e.s.
Valor: el de e
Efecto: **-f4-**escribir el valor de e (sin salto de línea)

(PRINT1 e)
Tipo: e.s.
Valor: el de e
Efecto: **-f4-**escribir el valor de e (con salto de línea)

(PROG (x1...xn) f1...fk)
Tipo: n.n.
Valor: el de fk o el de la primera forma RETURD que encuentre
Efecto: **-f4-**los de evaluar algunas de las f1,..., fk

(PROGN f1...fk)
Tipo: e.n.
Valor: el de fk
Efecto: **-f4-**los de evaluar f1,..., fk

(PUT id atr val)
Tipo: e.s.
Valor: el de id
Efecto: **-f4-**pone val como valor del atributo atr de id; o error, si ya existe atr en la lista de propiedades de id

(QUOTE s)
Tipo: n.n.
Valor: la S-expresión s (sin evaluar)

(READ)
Tipo: n.n.
Valor: la siguiente S-expresión en el dispositivo de entrada
Efecto: **-f4-**leer del dispositivo de entrada.

(RECLAIM)
Tipo: n.n.
Valor: NIL
Efecto: **-f4-**provoca una recogida de basura en la memoria de trabajo

(REMOVE e)
Tipo: e.s.
Valor: NIL
Efecto: **-f4-**elimina el omo e de la lista de objetos

(REMPROP id atr)
Tipo: e.s.
Valor: el valor del atributo atr en la lista de propiedades de id; NIL, si no existe ese atributo.
Efecto: **-f4-**limina el atributo y su valor de la lista de propiedades de id; un mensaje de aviso si no existiera el atributo

(RETURN e)
Tipo: e.s.
Valor: el de e
Efecto: **-f4-**termina la ejecución de una forma prog

(REVERSE lis)
Tipo: e.s.
Valor: la lista lis en orden inverso

(RPLACA e1 e2)
Tipo: e.s.
Valor: el de e1
Efecto: **-f4-**reemplaza el CAR de e1 por el valor de e2

(SET e1 e2)
Tipo: e.s.
Valor: el de e2
Efecto: **-f4-**liga al valor de e1 con el valor de e2; da error si e1 no evalúa a una variable declarada o a un parámetro formal

(SETQ id e)
Tipo: n.n.
Valor: el de e
Efecto: **-f4-**liga a id el valorde e; da error si id no es una variable declarada o un paretro formal

(SPACES n)
Tipo: e.s.
Valor: el de n, que debe ser un número
Efecto: **-f4-**escrbie n caracteres blancos (sin salto de líneas)

(SUB e1 e2)
Tipo: e.s.
Valor: la diferencia entre e1 y e2

(TERPRI)
Tipo: n.n.
Valor: NIL
Efecto: **-f4-**produce un salto de línea en el dispositivo de salida

(TRACE 1)
Tipo: e.s.
Valor: la lista de las funciones a trazar
Efecto: **-f4-**une 1 a la lista de funciones a trazar

(UNTRACE 1)
Tipo: e.s.
Valor: la lista de funciones a trazar
Efecto: **-f4-**borra de la lista de funciones a trazar los elementos de 1

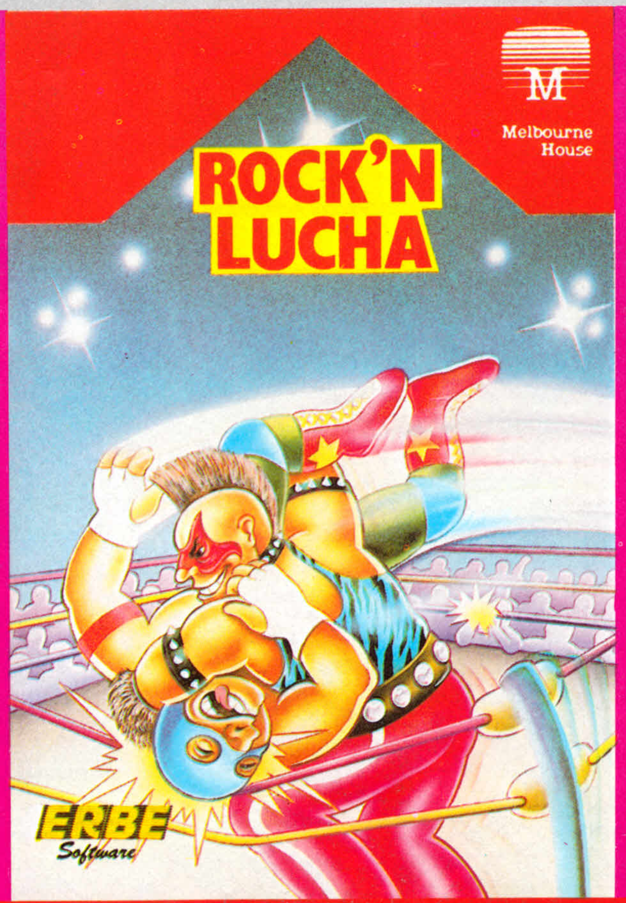
(^ e1 e2)
Tipo: e.s.
Valor: e1 elevado a e2

AMSTRAD

ERBEdades

SABOTEUR

Como experimentado mercenario cuidadosamente entrenado en las artes marciales, debes cumplir la misión que te ha sido encomendada: robar el disco que con la información de los rebeldes tiene el Gran Dictador.



ROCK'N LUCHA

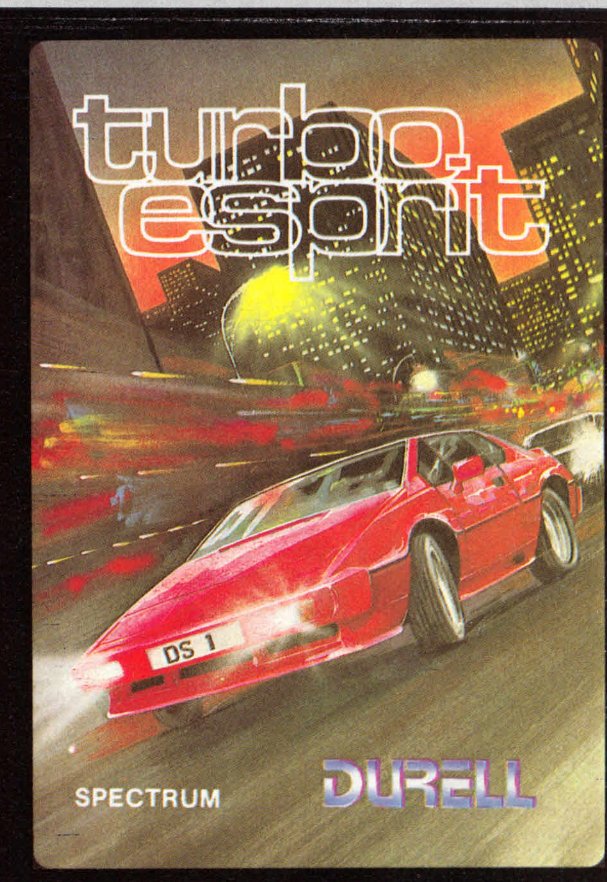
El primer juego de lucha libre hecho para ordenador. Más de 25 movimientos diferentes te permitirán hacer todo tipo de llaves: desde la sujección de espaldas hasta la voltereta de hombros, pasando por los mismos programadores del legendario "Exploding Fist".

THE WAY OF THE TIGER

Entra en el mudo de los samurais. Mantén la calma mientras el movimiento y las rutinas de combate te transportan a niveles que nunca pensaste posibles. Experimenta los sorprendentes efectos del "Triple Scroll" mientras intentas mejorar tus técnicas de lucha cuerpo a cuerpo, con espada samurai o con mil posibilidades más.

RAMBO

Toda la emoción del film, en tu ordenador. Siéntete como John Rambo en la jungla vietnamita e intenta salvar a tus compañeros prisioneros en el campo de concentración.



TURBO ESPRIT

Tu misión: vigilar y cuidar el cumplimiento de la ley que se ve amenazada por una terrible banda de delincuentes que han hecho del tráfico de narcóticos su negocio más rentable.
Tus medios: un Lotus Turbo Sprit dotado de uno de los máximos adelantos técnicos y con el que deberás patrullar por calles y avenidas.

GUN-FRIGHT

Ultimate nos ofrece para Amstrad, usando su técnica Filmation, "Gun-Fright", el juego en el que el lejano oeste es el protagonista.
Ponte en el papel de Quikdraw, el sheriff que piensa librar la ciudad de todos los pistoleros a lomos de su buen caballo Panto.

REUBICADOR DE PROGRAMAS EN C/M

La REUBICACION de programas en código máquina se divide en dos partes. Primero uno tiene que escribir el programa en forma REUBICABLE. Entonces uno lo presenta al sistema con un cargador de REUBICAR, el cual se ocupa de colocar el nuevo código junto a cualquier memoria ya en uso, y deja el sistema tan poco cambiado como sea posible.

Una rutina que se reubica por sí mismo tiene la marca de ser profesional. Una que no lo hace es semi-profesional. Para nuestros propios usos en casa todos los programadores hacemos la reubicación en el momento de ENSAMBLAR: variando ORG simplemente. Pero es como usar una manivela para arrancar el coche en lugar de la batería. **Reubicación es la batería de arranque que quita los problemas en el momento de poner el sistema en marcha.** No obstante, la reubicación es el toque final a un programa, y es lo suficientemente difícil (hablando del Z80) que no suele aplicarse hasta que el programa esté en su forma definitiva.

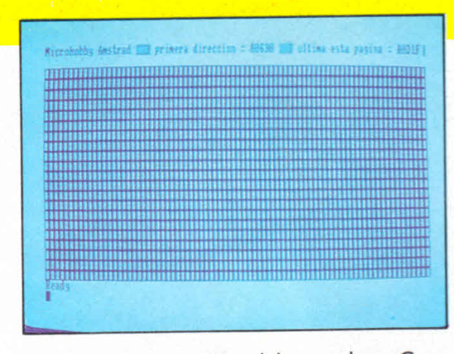
Rutinas en máquinas profesionales

La rutina que transforma un programa en reubicable se ve en el listado de ensamblador. Es el mismo que hemos usado en el programa HOBBYCOP del número 27 de la revista, pero entonces figuraban sólo los bytes del código máquina. El uso de ORG 0 hace que todas las etiquetas generadas en la compilación son relativas al principio de la rutina. La llamada falsa al firmware fija el valor del PC (Program Counter) que se recupere del stack y se añade a las direcciones relativas necesarias en la manera mostrada. El uso de una llamada falsa para fijar PC, como a menudo pasa con programación del Z80, es un truco que una vez visto, no se olvida.

Generación de la tabla

¿Cómo se genera la tabla de reubicación? El programa HOJEAMEN.TXT se presenta con su listado en el momento de ensamblar. Se ha usado GENA de Hisoft —un ensamblador de dos pasos— que obliga a colocar la tabla de reubicación al final. Se ha usado mayúsculas para las etiquetas que representan nudos lógicos del programa. Minúsculas se usan para puntos de entrada desde el programa principal *(p. ej. Basic)* y para las entradas en la tabla de reubicación. Estas últimas se ha dado un número «qq» si corresponde a instrucciones de tres bytes, y «**rr**» si son de cuatro. Sus entradas correspondientes en la tabla son entonces qq+1 y rr+2. No se reubica las instrucciones son saltos relativos: JR DJNZ, etc. Tampoco las llamadas al Jump Block *(saltos)* del firmware: entonces éstas se han dejado en forma numérica JP #BB33 en lugar de dar el nombre JP km.set.control.

Después de preparar la tabla de reubicación al final del programa, hay que incluirlo en la rutina de reubicar con algo como el comando *F HOJEAMEM.TXT de GENA o incluirlo directamente con un merge. Para los que usamos cassette, la segunda manera es la más fácil, siempre que la memoria lo permite.



Ahora se ensambla todo. Con ORG 0 el código objeto hay que guardarlo en alguna otra parte de la memoria. GENA lo hace con opción 16. Toma nota de la longitud de la TABLE al final del programa. Esta longitud se pone ahora en la pseudo-instrucción **«tabla: DEFS 76»** — 76 era la longitud para el ejemplo final, pasando los bytes de la **«TABLE»** a su nueva posición **«tabla»** para que no ocupen memoria indebidamente después de ser usados. Serán tirados con la basura. Una manera de hacerlo se da al final de la rutina de reubicar.

Programa HOJEAMEN

Un ejemplo de un CARGADOR de reubicar se ve en el programa Basic en el cual el fichero binario del ensamblador ha sido pasado a código máquina en líneas DATA. Claro que si tiene el fichero en binario, no hay que teclear líneas 100-350 ni la subrutina 2000 (2000-2999). En su lugar, linea 1007 se cambia a:

1007 LOAD''hojeamen'', h+1

y el binario se carga en memoria más rápidamente. **Si tecleas los DATA, usa MODE 2:** el formato es de 64 caracteres por línea para ajustarse a la norma de tu revista, y si saltas un dato será claro a simple vista. Hay un checksum al final de cada línea: un error se señala con el mensaje «Error en xxx» y las líneas de DATA listarán. No hace falta contar espacios: las identaciones (p. eje. líneas 2010-2110) se usan para ayudar comprender la estrucutra de los bucles, y no necesitas seguirlas. Las observaciones (REMs) que siguen al apóstrofe (') pueden reemplazarse con un par de asteriscos ('**').

Si no tienes interés ahora mismo en usar la reubicación, teclea el programa en Basic porque el resultado es una herramienta potente en abrir programas en binario, en seguir la disponibilidad de la memoria, en depurar programas largos en Basic, etc.

```
*H Rutina de reubicar : Microhobby AMSTRAD Semanal

        ORG 0000          ;usar opción-16 con GENA
GETLOC:EQU #BB21          ;llamada falsa al firmware
REUBIC:CALL GETLOC        ;pone PC+3 en el stack
        DEC SP
        DEC SP
        EX (SP),HL        ;HL=PC+3 original
        DEC HL
        DEC HL
        DEC HL            ;offset (constante de reubicar)
        INC SP
        INC SP            ;SP restaurado
        PUSH HL
        LD DE,tabla-2     ;tabla de reubicar estará aqui
        ADD HL,DE
        POP BC            ;BC=offset
REUB1  :LD E,(HL)         ;empezar bucle
        INC HL
        LD D,(HL)
        INC HL            ;saca una dirección de la tabla
        LD A,D
        OR E              ;comprobar señal para fin de tabla
        JR Z,REUB2        ;hecho
        PUSH HL           ;puntero en la tabla
        EX DE,HL          ;HL=dirección
        ADD HL,BC
        PUSH HL           ;dirección+offset
        LD E,(HL)
        INC HL
        LD D,(HL)
        EX DE,HL          ;HL=(dirección+offset)
        ADD HL,BC         ;offset añadido
        EX DE,HL
        POP HL            ;saca dirección+offset
        LD (HL),E         ;pokear el nuevo valor
        INC HL
        LD (HL),D
        POP HL            ;saca puntero en la tabla
        JR REUB1          ;al bucle
REUB2  :JP keydef         ;JP LOGRSX si es programa RSX ó RET
        DEFW REUB2+1      ;comenzar ya la reubicación
tabla  :DEFS 76           ;ó lo que sea, 76 en este ejemplo
endtbl:DEFW 0             ;señal de fin de tabla
        DEFS 2            ;zona de seguridad
*F HOJEAMEM.TXT

;ASSEMBLE opción-16 (estilo GENA)
;SAVE fichero binario en cinta o disco
;LOAD este mismo fichero a, digamos, #5000
;INTELLIGENT COPY bajo MONA (opción I)
;  Start: 5000+TABLE
;   Last: 5000+zzzzzz-1
;     To: 5000+tabla
;SAVE binario bajo MONA con opción WRITE (opción W)
;   Name: HOJEAMEM  por ejemplo
;  First: 5000
;   Last: 5000+TABLE (no +TABLE-1 un error en MONA)
;  Start: -
;Usar ajustandolo a h = himemory con LOAD a (h-TABLE)
;Tirar la basura de reubicación moviendo el puntero
;          de memoria arriba hasta (h-TABLE+entry).
;Versión: 7.ii.86
```

La rutina capta-error 5000 es un intento de dejar SYMBOL AFTER de BAsic en su estado original. Se nota que la memoria tira hacia arriba otra vez en 3000 después de la reubicación.

Listing HOJEAMEN.BAS

El programa que ahora tienes, HOJEAMEN.BAS, es diseñado a ser cargado cuando se enciende la máquina. Deja dos nuevas teclas de función: CTRL/ [.]-numérico te permite hojear por la RAM, y CTRL/ENTER-grande (o CTRL/RETURN estilo 128) te devuelve cuando lo quieres tu pantalla que estabas usando antes (INKs PAPER PEN BORDER MODE). Están asignados los números 158/159, los más altos posibles, para no molestar a otros programas que se puedan usar después. **Las rutinas están disponibles como teclas y no como RSX porque son de uso en modo comando y no en tiempo de ejecución.**

```
Tiene instalados los RSX's:
      |copy  |copylat
      |turboload  |turbosave   |cab

    y las teclas de funcion
      ctrl+ENTER grande (RETURN)=
                vuelta a pantalla normal
      ctrl+(.)numerico=
                hojear memoria

 Para sacar una copia de este programa,
 rebobinar la cinta y copiar el comando
        CHAIN"linkload",50000

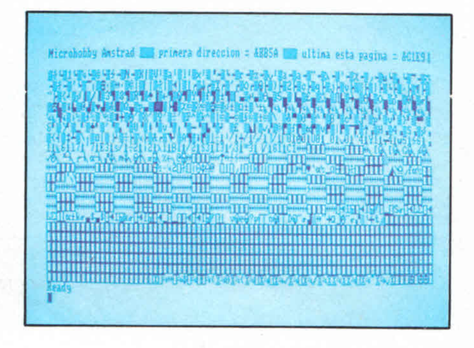
Ready
```

Ahora pulsa CTRL [.]-numérico. Dirección de comienzo: ENTER, y empieza en el 0000 por defecto. Estas son las celdas de la memoria. Es tan rápida la pantalla, que hay una pausa de dos segundos al final de cada página. Cuando se aburre de ver RAM vacía, pulsa una tecla y se parará. Ahora dirección de comienzo: &b446 y pulsa una tecla para pararlo ya. Lo que se ve es el buffer de las teclas de función con RUN'', nuestros CALL &XXXX, etc. (En modelos más modernos que el nuestro, puede ser que la dirección no es exactamente &B446, pero estará cercana.)

Cuando quieres, pulsa ESC una vez para volver a Basic. ¡Qué bonito! ¿Verdad? Si tienes impresora, puedes sacar lo que ves a papel usando nuestro RSX COPY de hace unas pocas semanas. (Era también reubicable.) Ahora pulsa CTRL/ENTER-grande y ¡voila!

No hay que quitar estas cosas de la máquina. Están allí para usarse con otros programas. Por ejemplo, estás tecleando un programa largo en Basic y algo no funciona. Hojea la memoria baja hasta que veas el final del Basic. Entonces se ve como Basic apila los variables al final de sí mismo durante la ejecución. Y se ve **"COUNT. SWAP..."** y muy cercana **"COUNT. SUAP..."** ¡Ajá! Un pequeño error de ortografía inglesa.

Vas a encontrar la función CTRL/ENTER-grande muy útil. Muchas veces nos encontramos con una pantalla ilegible. Tocamos esta tecla, y otra vez tenemos la pantalla del momento en que hicimos la última llamada a HOJEAMEN. Diviértete.



```
                  *L+ HOJEAMEM Amstrad Semanal 1Feb86
                  ;hojear memoria = ctrl/num.pad [.]
                  ;reponer inks,mode,paper,border,etc.
                  ;                = ctrl/ENTER (RETURN)
007F  3ADC00      keydef: LD   A,(DIRMES+6)
0082  EE24                XOR  36
0084  FE4C                CP   "L"
0086  C0                  RET  NZ
0087  069F                LD   B,159
0089  3E12                LD   A,18
008B  CD33BB              CALL #BB33
008E  05                  DEC  B
008F  3E07                LD   A,7
0091  C333BB              JP   #BB33
0094  CFCFA4      entry:  DEFB 207,207,164
0097  31393836            DEFM "1986 Hobby Press S.A."
00AC  0D12FF      ASKDR1: DEFB 13,18,#FF
00AF  07          ASKDIR: DEFB 7
00B0  496E7075            DEFM "Input direccion "
00C0  64652063            DEFM "de comienzo &...."
00D1  08080808            DEFB 8,8,8,8,#FF
00D6  0C          DIRMES: DEFB 12
00D7  4D696372            DEFM "Microhobby Amstrad "
00EA  CFCFCF              DEFB 207,207,207
00ED  20707269            DEFM " primera direccion = &"
0103  FF                  DEFB #FF
0104  20CFCFCF    DIRMS2: DEFB 32,207,207,207
0108  20756C74            DEFM " ultima esta pagina = &"
011F  FF                  DEFB #FF
0120  D90AFF      DIRMS3: DEFB 217,10,#FF
0123              DIRECC: DEFS 2
0125  00          USFLAG: DEFB 0
0126  0D1C20      USDATA: DEFB 13,28,32
0129  0B0B1C01    USINKO: DEFB 11,11,28,1
012D  202004      USINK1: DEFB 32,32,4
0130  021D        USMOD:  DEFB 2,29
0132  0B0B0E      USBOR:  DEFB 11,11,14
0135  200F        USPAP:  DEFB 32,15
0137  010DFF      USPEN:  DEFB 1,13,#FF
013A  0D1C20      BRDATA: DEFB 13,28,32
013D  0B0B1C01    brink0: DEFB 11,11,28,1
0141  20200402    brink1: DEFB 32,32,4,2,29
0146  0B0B0E20    brbor:  DEFB 11,11,14,32,15,1,13,7,#FF
014F              reinic:
014F  AF                  XOR  A
0150  322501      qq1:    LD   (USFLAG),A
0153  212601      qq2:    LD   HL,USDATA
0156  C3F501      qq3:    JP   PRTTXT
0159              browse:
0159  3A2501              LD   A,(USFLAG)
015C  B7                  OR   A
015D  2036                JR   NZ,BROWS1
015F  CD93BB              CALL #BB93
0162  323701      qq4:    LD   (USPEN),A
0165  CD99BB              CALL #BB99
0168  323501      qq5:    LD   (USPAP),A
016B  CD11BC              CALL #BC11
016E  323001      qq6:    LD   (USMOD),A
0171  AF                  XOR  A
0172  CD35BC              CALL #BC35
0175  ED432901    rr1:    LD   (USINKO),BC
0179  3E01                LD   A,1
017B  322501      qq7:    LD   (USFLAG),A
017E  CD35BC              CALL #BC35
0181  ED432D01    rr2:    LD   (USINK1),BC
0185  CD3BBC              CALL #BC3B
0188  ED433201    rr3:    LD   (USBOR),BC
018C  213A01      qq8:    LD   HL,BRDATA
018F  CD19BD              CALL #BD19
0192  CDF501      qq9:    CALL PRTTXT
0195  21AC00      BROWS1: LD   HL,ASKDR1
0198  CDF501      qq10:   CALL PRTTXT
019B  CD81BB              CALL #BB81
019E  CDFF01      BROWS6: CALL KFLUSH
01A1  21AF00      qq11:   LD   HL,ASKDIR
01A4  CDF501      qq12:   CALL PRTTXT
01A7  0604                LD   B,4
01A9  210000              LD   HL,0
01AC  222301      qq13:   LD   (DIRECC),HL
01AF  212301      qq14:   LD   HL,DIRECC
01B2  CD06BB      INPHEX: CALL #BB06
01B5  FEFC                CP   #FC
01B7  2839                JR   Z,BROWS5
01B9  FE0D                CP   13
01BB  2805                JR   Z,INPHX1
01BD  CD0802      qq15:   CALL ASC2HL
01C0  20F0                JR   NZ,INPHEX
01C2  CDFF01      INPHX1: CALL KFLUSH
01C5  ED5B2301    rr4:    LD   DE,(DIRECC)
01C9  CD84BB      BROWS2: CALL #BB84
01CC  CD2902      qq16:   CALL DIROUT
01CF  0E07                LD   C,7
01D1  06F0        BROWS4: LD   B,240
01D3  1A          BROWS3: LD   A,(DE)
01D4  C5                  PUSH BC
01D5  D5                  PUSH DE
01D6  CD5DBB              CALL #BB5D
01D9  D1                  POP  DE
01DA  C1                  POP  BC
01DB  13                  INC  DE
01DC  10F5                DJNZ BROWS3
01DE  0D                  DEC  C
01DF  20F0                JR   NZ,BROWS4
01E1  CD81BB              CALL #BB81
01E4  3EC8                LD   A,200
01E6  CD6702      qq17:   CALL CDELAY
01E9  CD1BBB              CALL #BB1B
01EC  30DB                JR   NC,BROWS2
01EE  FEFC                CP   #FC
01F0  20AC                JR,  NZ,BROWS6
01F2  21AC00      BROWS5: LD   HL,ASKDR1
01F5              PRTTXT:
01F5  7E                  LD   A,(HL)
01F6  23                  INC  HL
01F7  FEFF                CP   #FF
01F9  C8                  RET  Z
01FA  CD5ABB              CALL #BB5A
01FD  18F6                JR   PRTTXT
01FF  0600        KFLUSH: LD   B,0
0201  CD09BB      KFL1:   CALL #BB09
0204  D0                  RET  NC
0205  10FA                DJNZ KFL1
0207  C9                  RET
0208  4F          ASC2HL: LD   C,A
0209  FE61                CP   "a"
020B  3802                JR   C,ASCHL1
020D  D620                SUB  #20
020F  D630        ASCHL1: SUB  "0"
0211  D8                  RET  C
0212  FE0A                CP   10
0214  3808                JR   C,ASCHL2
0216  D607                SUB  7
0218  FE0A                CP   10
021A  D8                  RET  C
021B  FE10                CP   16
021D  D0                  RET  NC
021E  ED6F        ASCHL2: RLD
0220  23                  INC  HL
0221  ED6F                RLD
0223  2B                  DEC  HL
0224  79                  LD   A,C
0225  05                  DEC  B
0226  C35ABB              JP   #BB5A
0229  21D600      DIROUT: LD   HL,DIRMES
022C  CDF501      qq18:   CALL PRTTXT
022F  4A                  LD   C,D
0230  CD4F02      qq19:   CALL CHEXPT
0233  4B                  LD   C,E
0234  CD4F02      qq20:   CALL CHEXPT
0237  210401      qq21:   LD   HL,DIRMS2
023A  CDF501      qq30:   CALL PRTTXT
023D  218F06              LD   HL,240*7-1
0240  19                  ADD  HL,DE
0241  4C                  LD   C,H
0242  CD4F02      qq31:   CALL CHEXPT
0245  4D                  LD   C,L
0246  CD4F02      qq32:   CALL CHEXPT
0249  212001      qq33:   LD   HL,DIRMS3
024C  C3F501      qq22:   JP   PRTTXT
024F  79          CHEXPT: LD   A,C
0250  E6F0                AND  #F0
0252  0F                  RRCA
0253  0F                  RRCA
0254  0F                  RRCA
0255  0F                  RRCA
0256  CD5C02      qq23:   CALL NASCII
0259  79                  LD   A,C
025A  E60F                AND  #0F
025C  FE0A        NASCII: CP   10
025E  3802                JR   C,NAS1
0260  C607                ADD  A,7
0262  C630        NAS1:   ADD  A,"0"
0264  C35ABB              JP   #BB5A
0267  01D809      CDELAY: LD   BC,2520
026A  0D          CDLAY1: DEC  C
026B  20FD                JR   NZ,CDLAY1
026D  05                  DEC  B
026E  20FA                JR   NZ,CDLAY1
0270  3D                  DEC  A
0271  20F4                JR   NZ,CDELAY
0273  C9                  RET
0274  80005101    TABLE:  DEFW keydef+1,qq1+1,qq2+1,qq3+1
027C  5A016301            DEFW browse+1,qq4+1,qq5+1,qq6+1
0284  77017C01            DEFW rr1+2,qq7+1,rr2+2,rr3+2
028C  8D019301            DEFW qq8+1,qq9+1,BROWS1+1,qq10+1
0294  9F01A201            DEFW BROWS6+1,qq11+1,qq12+1,qq13+1
029C  B001BE01            DEFW qq14+1,qq15+1,INPHX1+1,rr4+2
02A4  CD01E701            DEFW qq16+1,qq17+1,BROWS5+1
02AA  2A022D02            DEFW DIROUT+1,qq18+1,qq19+1,qq20+1
02B2  38024D02            DEFW qq21+1,qq22+1,qq23+1
02B8  3B024302            DEFW qq30+1,qq31+1,qq32+1,qq33+1
02C0              zzzzzz:

Pass 2 errors: 00
```

```
100 DATA CD21BB3B3BE32B2B2B3333E5112D0019C15E2356237AB3281108 46
```

```
100 DATA CD21BB3B3BE32B2B2B3333E5112D0019C15E2356237AB32811084 6
```

Ofites Informática

Presenta:
el lápiz al que gusta decir SI
mientras nuestros competidores dicen no

UNICO PARA AMSTRAD, CON PRECISION PIXEL

FUNCIONES	ESP	dk'tronics	OTROS
UNICO MENU DE PANTALLA	SI	NO	
ARRASTRE OBJETOS PANTALLA	SI	NO	
TRASLADO OBJETOS PANTALLA	SI	NO	
TRASLADO DE CURSOR	SI	NO	
CAJAS ELASTICAS	SI	SI	
LINEA ELASTICA	SI	SI	
TRIANGULO ELASTICO	SI	NO	
ELIPSE ELASTICO	SI	NO	
DIAMANTE ELASTICO	SI	NO	
POLIGONO ELASTICO	SI	NO	
HEXAGONO ELASTICO	SI	NO	
OCTOGONO ELASTICO	SI	NO	
CUBO ELASTICO	SI	NO	
PIRAMIDE ELASTICA	SI	NO	
CIRCUNFERENCIAS	SI	SI	
CIRCULOS RELLENOS	SI	NO	
CAJAS RELLENAS	SI	NO	Compare con otros lápices
ELIPSES RELLENAS	SI	NO	
CUÑAS	SI	NO	
SIMULADOR DE CORTES	SI	NO	
DISEÑO DE ZOOM	SI	SI	
IMAGEN ESPEJO E INVERTIDA	SI	NO	
FONDO DE REFERENCIA	SI	NO	
REJILLA DE FONDO	SI	NO	
OPCION DISPLAY X, Y	SI	NO	
RELLENADO CON COLOR	SI	SI	
LAVADO DE COLOR	SI	NO	
VOLCADO PANTALLA RESIDENTE	SI	NO	
DIBUJO DE BORDES EN 3 D	SI	NO	
TEXTO	SI	SI	
9 TAMAÑOS DE BROCHA	SI	NO	
18 TOBERAS MOSTRADORAS	SI	NO	
4 MEZCLAS BASICAS	SI	NO	
VARIADOR DE MEZCLAS	SI	NO	
SOMBREADO DE MEZCLAS XOR	SI	NO	
FICHERO ICONOS RESIDENTES	SI	NO	
FICHERO RELLENOS RESIDENTES	SI	NO	
26 COLORES DE PAPEL	SI	NO	
PALETA DE 15 TONOS DE COLOR	SI	NO	
POSICIONAMIENTO DE PUNTO	SI	SI	
RAYOS DESDE UN PUNTO FIJO	SI	NO	
DIBUJO REFLEJADO (ESPEJO)	SI	NO	
FUNCION HOME	SI	NO	
CONTROL DESDE TECLADO	SI	SI	
CONTROL CON JOYSTICK	SI	NO	
DISPONIBLES MODOS 1 Y 2	SI	?	

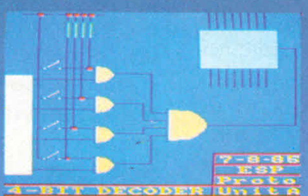
DEBIDO A LA FALTA DE ESPACIO NO PODEMOS LISTAR LAS OTRAS 40 FUNCIONES MAS QUE NUESTRO LAPIZ ES CAPAZ DE HACER.

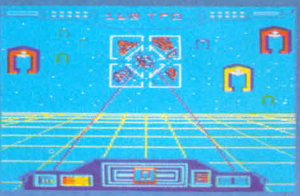


ESTOS SON ALGUNOS EJEMPLOS DE LOS GRAFICOS QUE VD. PODRA REALIZAR CON NUESTRO LAPIZ OPTICO









DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

DISPONIBLE PARA:

CPC 464 CASSETTE	4.900 Ptas.
CPC 464-664 DISCO	6.900 Ptas.
CPC 6128 DISCO	6.900 Ptas.

(IVA no incluido)

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

Si Vd. tiene alguna dificultad para obtener el lápiz óptico, puede dirigirse a:



Avda. Isabel II, 16 -8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

FIDICON

En los últimos años el concepto de programa ha derivado al de paquete integrado. Nuestro Amstrad tiene capacidad y potencia para soportar uno de estos paquete, pero el precio prohibitivo de éstos nos ha obligado a hacernos el nuestro.

La estructura de FIDICON es muy simple, pero cuando hacemos RUN «FIDICON» empiezan las sorpresas, nos pregunta: «¿QUE PROGRAMA DESEA UTILIZAR?» (!) Apareciendo cuatro opciones. La cuarta opción nos resulta familiar **«FIN DE OPERACIONES»**, cuidado con ella, no pide confirmación. Pero, y las otras tres. Son los programas que hemos cargado en nuestro **Amstrad**, cuando elegimos uno de los tres los otros dos actúan como subrutinas de éste, así podemos mantenerlos en memoria sin que uno borre a otro. Desventaja quienes trabajen con cinta tardarán más tiempo en cargar, por lo que es recomendable que las rutinas que definen a uno y otro programa las graben en otra cinta y hagan un margen con la que vayan a utilizar.

Prosigamos, tras elegir un programa nos encontraremos con la descripción del formato utilizado por el programa y algunas observaciones. Tras esto responderemos a las siguientes cuestiones con **«D»** o **«C»** y al número de registros, de acuerdo con el número de éstos que vamos a tratar. Si ponemos de menos tendremos que grabar el fichero y redimensionar el programa.

Nos encontramos ahora con un menú con opciones que nos son familiares, **CARGAR FICHERO**, elegida esta opción se nos preguntará si queremos ver el catálogo del disco. En todo momento sabremos qué registro se está cargando, una vez finalizada la carga volveremos al menú principal.

CREAR FICHERO, iremos rellenando los campos, cuidando de no sobrepasar el número de caracteres definidos, pues aparecerá recortado al listarlo. Introduciremos «*» para salir.

CONSULTAR O LISTAR FICHERO, nos hará nuevamente preguntas, y dispondremos de una opción que dota de gran potencia al programa, **«BUSQUEDA POR ABREVIATURA?»**, con esta opción dispondremos de la posibilidad de listar todos los Fernández o García que haya en nuestro dietario, o todas las compras del mes de marzo, si habíamos seguido la recomendación del programa de dar la fecha de la forma MM (mes)/DD (día)/AA (a/o), pidiendo el listado por la clave «MM» o las compras de un día del mes introduciendo »MM/DD», practique con esta opción.

Una vez que hemos respondido todas las cuestiones nos aparecerá el listado por pantalla o impresora y en el monitor aparecerá una ventana con unas opciones: **RETORNO MENU**,

NUEVO REGISTRO: nos permite introducir nuevos registros a partir de la posición que le indiquemos, borrando los que haya debajo, si los hay, por medio del teclado o realizando un merge desde cinta o disco.

BAJAS, introduciremos la posición del registro a borrar y el programa se encargará de compactar el fichero evitando así que queden huecos.

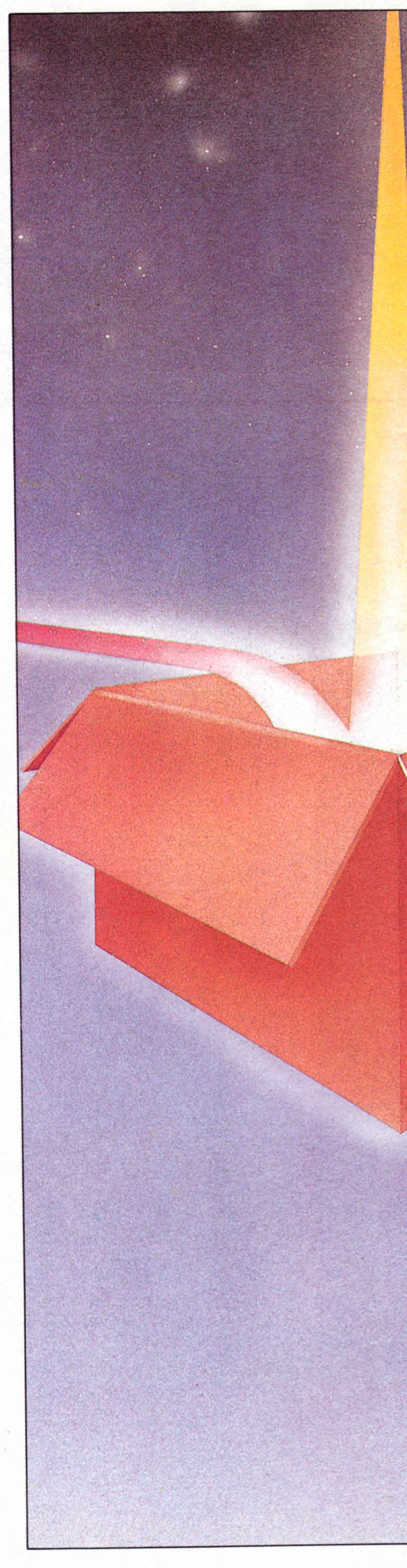
BAJAS, introduciremos la posición del registro a borrar y el programa se encargará de compactar el fichero evitando así que queden huecos.

MODIFICAR, nos pedirá la posición del registro a modificar, y la del campo, tras la modificación en el programa contabilidad se actualizará el campo saldo si fuera necesario mostrando un listado por pantalla o impresora.

PAG. ANT. vuelve atrás una página, para evitar tener que introducir de nuevo las opciones que los seleccionaron.

CONTINUA, sigue adelante con el listado tanto por pantalla como por impresora.

De nuevo en el menú principal, la siguiente opción es ORDENAR POR CLAVES, una vez elegida la clave, el programa nos indicará constantemente de cuántos registros van orde-





nados y cuántos faltan por ordenar, regresando luego al menú principal.

SALVAR FICHERO, de la misma forma nos indicará los registros salvados.

REGRESAR MENU DE PROGRAMAS, pide confirmación ya que repondiendo que sí, borraremos el fichero en memoria.

BORRAR FICHERO DE MEMORIA, al igual que la anterior debe manejarse con cuidado.

Los ficheros creados por el programa dietario, pueden ser tratados por el programa contabilidad, lo cual nos facilitará obtener el saldo de un período de trabajo.

```
 10 MODE 1:INK 0,0:PAPER 0:CLS:INK
1,10:PEN 1:BORDER 0
20 '********************************
******************************
30 '******         programmed by JOSE
TORO 21-2-86               *******
40 '******
                           *******
50 '****** realizado sobre un cpc-4
64 para AMSTAD SEMANAL *******
60 '******
                           *******
70 '******          compatible 464,47
2,664,6128                 *******
80 '********************************
******************************
90 CALL &BB9C:LOCATE 1,8:PRINT SPAC
E$(40):LOCATE 1,16:PRINT SPACE$(40)
100 CALL &BB9C
110 LOCATE 17,11:PRINT"FIDICON":LOC
ATE 10,14:PRINT"PROGRAMMED BY JOSE
TORO"
120 FOR U=1 TO 1000:NEXT
130 CLEAR:MODE 2 :INK 0,27:INK 1,3:
PEN 1:PAPER 0:BORDER 0:CLS
140 '<<<<<<<<<<<<<<<FIDICON=FICHERO
+DIETARIO+CONTABILIDAD >>>>>>>>>>>>
>>>>>
150 '********************** ELECCION
 DE PROGRAMA **********************
160 MODE 2:SOUND 1,20,30,5,0
170 LOCATE 8,20:PRINT"QUE PROGRAMA
DESEA UTILIZAR:"
180 LOCATE 30,2:PRINT"OPCIONES":LOC
ATE 30,3:PRINT"========"
190 LOCATE 18,5:PRINT"1. FICHERO DE
 ACCESO MULTIPLE"
200 LOCATE 18,8:PRINT"2. DIETARIO"
210 LOCATE 18,11:PRINT"3. CONTABILI
DAD"
220 LOCATE 18,14:PRINT"4. FINALIZAR
 OPERACIONES"
230 LOCATE 38,20:CLEAR:prog$="":WHI
LE prog$="":prog$=INKEY$:WEND:prog=
VAL(prog$)
240 IF prog=4 THEN CALL bd19
250 IF prog<0 OR prog>3 THEN 140
260 IF prog=1 THEN jota=7 ELSE jota
=6
1000 '*********** presentacion *****
************
1010 MODE 2 :INK 0,27:INK 1,3:PEN 1
:PAPER 0:BORDER 0:CLS
1020 ON prog GOSUB 26000,27000,2800
0
1030 CLS:PRINT
1040 PRINT" DESEA UTILIZAR DISCO O
CINTA (d/c)":cas$="":WHILE cas$="":
cas$=INKEY$:WEND:cas$=UPPER$(cas$)
1050 PRINT
1060 n=1:s=0
1070 IF proo=1 THEN PRINT" Puede ut
ilizar 1000 registros como maximo (
si borra las lineas 20000-21420  y
    24000- podras utilizar 1400 reg
istros)."
1080 PRINT" Es recomendable dar un
margen en el numero de registros a
utilizar (con 30 de     mas sera suf
iciente)."
1090 PRINT:INPUT " INTRODUZCA NUMER
O DE REGISTROS, (maximo 1100) ",m:I
F m>1100 OR m<0 THEN 1010
1100 IF m=0 THEN m=1100
1110 DIM mat$(m,jota)
1120 CLS:MODE 1:BORDER 0:a$="":f=0:
CATA$=""
1130 PRINT:PRINT:PRINT"
OPCIONES":PRINT "             ======
==":PRINT
1140 PRINT:PRINT"          1. CARGAR F
ICHERO"
1150 PRINT:PRINT"          2. CREAR FI
CHERO"
1160 PRINT:PRINT"          3. CONSULTA
R O LISTAR FICHERO"
1170 PRINT:PRINT"          4. ORDENAR
POR CLAVES"
1180 PRINT:PRINT"          5. SALVAR F
ICHERO"
1190 PRINT:PRINT"          6. REGRESAR
 MENU DE PROGRAMAS"
1200 PRINT:PRINT"          7. BORRAR F
ICHERO DEL DISCO"
1205 PRINT:PRINT"          8. BORRAR F
ICHERO DE MEMORIA"
1210 SOUND 1,500,30,5,0
1220 LOCATE 10,24:PRINT "introduzca
 opcion ":WHILE a$="":a$=INKEY$:WEN
D:opc=VAL(a$):BORDER 10
1230 ON opc GOSUB 3000,2000,4000,50
00,6000,7000,8000,8500
1240 GOTO 1120
2000 '********** creacion de ficher
o ****************
2020 MODE 1:posic=1
2030 MODE 2:s=0
2040 ON prog GOSUB 15000,18000,2200
0
2050 PEN#3,1:PAPER#3,0:CLS#3:BORDER
 0
2060 v=0:IF s=1 THEN RETURN
2070 PRINT:WRITE "introduzca '#' pa
ra terminar"
2080 FOR i=posic TO m
2090 i$=STR$(i):j=1:mat$(i,j)=i$:PR
INT#j,mat$(i,j):FOR j=2 TO jota
2100 IF prog=3 AND j=5 THEN j=6
2110 INPUT#j,"",mat$(i,j)
2120 mat$(i,j)=UPPER$(mat$(i,j))
2130 IF mat$(i,j)="#" THEN 2180
2140 NEXT j
2150 IF i/20=INT (i/20) THEN FOR k=
1 TO jota:CLS#k:NEXT
2160 inc=i+1
2170 NEXT i
2180 RETURN
3000 '************ cargar fichero *
*****
3010 BORDER 0:posic=1:REC=0
3020 MODE 2:BORR=0:PRINT#2,:PRINT#2
,"                          CARGAR F
ICHERO"
3030 PRINT#2:PRINT#2,"
        DESEA VER CATALOGO DEL DISC
O":WHILE CATA$="":CATA$=INKEY$:WEND
:CATA$=UPPER$(CATA$)
3040 WINDOW#0,1,80,5,20
3050 IF cas$="D" AND CATA$="S" THEN
 CAT
3060 LOCATE#2,12,20:PRINT#2,"   int
roduzca '#' para regresar al menu":
LOCATE#2,8,24
3070 CALL &BB03:INPUT#2,"          IN
TRODUZCA NOMBRE DEL FICHERO ";FICH$
:IF FICH$="#" THEN RETURN
3080 IF BORR=1 THEN RETURN
3090 IF REC=1 THEN OPENOUT FICH$ EL
SE OPENIN FICH$
3100 CLS
3110 FOR i=posic TO m
3120 FOR j=1 TO jota
3130 IF REC=1 THEN PRINT#9,mat$(i,j
):IF mat$(i,j)="#" THEN 3200
3140 IF REC=0 THEN INPUT#9,mat$(i,j
):IF mat$(i,j)="#" THEN 3200
3150 IF LEN(mat$(i,j))=0 THEN 3200
3160 mat$(i,j)=UPPER$(mat$(i,j))
3170 NEXT j
3180 LOCATE#1,25,13:IF REC=1 THEN P
RINT#1,"SALVANDO REGISTRO";i ELSE
PRINT#1,"CARGANDO REGISTRO";i
3190 NEXT i
3200 IF REC=1 THEN CLOSEOUT ELSE CL
```

```
OSEIN
3210 RETURN
4000 '*************** busca titulo
****
4010 MODE 2:BORDER 0:saldo=0
4020 CALL &BB9C:PRINT"EL LISTADO DE
L FICHERO SE CONSIGUE PULSANDO <CR>
 A TODAS LAS OPCIONES":CALL &BB9C:P
RINT
4030 PRINT "QUIERE LOS DATOS POR LA
 IMPRESORA "
4040 re$=INKEY$:IF re$="" THEN 4040
 ELSE IF re$="S" OR re$="s" THEN p=
8 ELSE p=1
4050 ordenar=0
4060 ON prog GOSUB 16050,20010,2401
0
4070 IF ordenar=1 THEN RETURN
4080 PRINT:PRINT:PRINT"INTRODUZCA N
UMERO DE CAMPO ";:A$="":WHILE A$=""
:A$=INKEY$:WEND:n=VAL(A$):PRINT A$:
PRINT
4090 PRINT"BUSQUEDA POR ABREVIATURA
 ";:ayuda$="":WHILE ayuda$="":ayuda
$=INKEY$:WEND:ayuda$=UPPER$(ayuda$)
:PRINT ayuda$
4100 PRINT:INPUT "INTRODUZCA CLAVE
A LISTAR (pulse <CR> a continuacion
) ",nombre$:nombre$=UPPER$(nombre$)
4110 ON prog GOSUB 16010,19010,2301
0
4120 k=1:vent=1:GOSUB 9000
4130 vent=0
4140 FOR i=k TO m
4150 j=1:IF LEN(mat$(i,j))=0 THEN 4
260
4160 IF mat$(i,2)="#" THEN 4280
4170 IF ayuda$<>"S" THEN 4200
4180 FOR l=1 TO 15
4190 IF nombre$=LEFT$(mat$(i,n),l)
OR LEFT$(nombre$,l)=mat$(i,n) THEN
4220 ELSE NEXT:GOTO 4260
4200 IF nombre$=mat$(i,n) THEN 4210
 ELSE 4260
4210 mat$(i,1)=STR$(i):IF prog=3 TH
EN saldo=VAL(mat$(i,4))+saldo:mat$(
i,5)=STR$(saldo)
4220 ON prog GOSUB 17010,21010,2501
0
4230 IF prog<>1 THEN IF i/10=INT(i/
10) THEN CLS#3:a$=INKEY$:IF a$="1"
THEN RETURN ELSE vent=0:GOSUB 9000:
GOTO 4250
4240 IF i/20=INT(i/20) THEN CLS#3:a
$=INKEY$:IF a$="1" THEN RETURN ELSE
 vent=0:GOSUB 9000
4250 IF opc=1 THEN RETURN
4260 NEXT i:L=0
4270 IF ordenar=2 THEN RETURN
4280 GOSUB 9000
4290 RETURN
5000 '********************** ORDENA
CION POR CLAVES *******************
***
5010 CLS:BORDER 0:PRINT:PRINT"
          CAMPOS":PRINT"
       ======"
5020 ordenar=1:GOSUB 4060
5030 PRINT:PRINT:PRINT:A$="":PRINT
"          SELECCIONE CAMPO A ORDENA
R":WHILE A$="":A$=INKEY$:WEND:CL=VA
L(A$):IF CL=0 OR CL>7 THEN RETURN
5040 CLS:BORDER 0
5050 FOR i=1 TO m:IF mat$(i,2)="#"
THEN con=i-1 ELSE NEXT i
5060 FOR i=1 TO con-1
5070 FOR j=con TO i+1 STEP -1
5080 IF mat$(i,CL)="" OR mat$(j,CL)
="" THEN 5100 ELSE LOCATE 1,10:PRIN
T"VAN ORDENADOS";i;"REGISTROS FALTA
N";con-i
5090 IF mat$(i,CL)>mat$(j,CL) THEN
FOR K=2 TO 6:B$(K)=mat$(i,K):mat$(i
,K)=mat$(j,K):mat$(j,K)=B$(K):NEXT
K
5100 NEXT j
5110 NEXT i
5120 RETURN
6000 '************ salvar fichero *
****
6010 MODE 2:REC=1:BORDER 0
6020 PRINT#2:PRINT#2,"
            AVISO!!!"
6030 WINDOW#0,1,80,5,20
6040 PRINT#2:PRINT#2," NO REPITA LO
S NOMBRES DE LOS FICHEROS DARIAN PR
OBLEMAS AL BUSCARLOS"
6050 CATA$="":GOSUB 3030
6060 RETURN
7000 '*************** FIN OPERACION
ES ********************************
*****
7010 MODE 1:BORDER 0:LOCATE 1,13:IN
PUT "QUIERE REGRESAR MENU DE PROGRA
MAS (s/n)";r$
7020 r$=UPPER$(r$):IF r$="S" THEN 1
0
7030 RETURN
8000 '***************** BORRAR FICH
ERO EN DISCO ********************
8010 MODE 2:BORDER 0
8020 IF cas$<>"D" THEN RETURN
8030 PRINT"
   ATENCION!!!"
8040 PRINT"
   ===========":PRINT
8050 BORR=1:PRINT:PRINT"    UTILICE
 ESTA OPCION CON CUIDADO, ASEGURESE
 ANTES DE SEGUIR":PRINT:PRINT"
       EL FICHERO BORRADO ES IRREC
UPERABLE"
8060 PRINT"
          ============="
8070 GOSUB 3050:IF FICH$="#" THEN R
ETURN
8080 |ERA,@FICH$:BORR=0
8090 RETURN
8500 '************** borrar fichero
 de memoria ***********
8510 CLS:INPUT "DESEA BORRAR EL FIC
HERO EXISTENTE EN MEMORIA (s/n)";a$
8520 IF a$<>"s" OR a$<>"S" THEN RET
URN ELSE CLEAR:RETURN
9000 '<<<<<<<<<<<<<< VENTANA DE AYU
DA >>>>>>>>>>>>>>>
9010 PEN#3,0:PAPER#3,1:CLS#3
9020 PRINT#3:PRINT#3,"1.RETORNO MEN
U  2.NUEVO REGISTRO  3.BAJAS  4.MOD
IFICAR  5.PAG.ANT.  6.CONTINUA"
9030 IF vent=1 THEN RETURN
9040 LOCATE#3,28,3:PRINT#3,"introdu
zca opcion ":A$="":WHILE A$="":A$=I
NKEY$:WEND:opc=VAL(A$)
9050 IF opc=1 THEN RETURN
9060 ON opc GOSUB 10,13000,10000,11
000,12000,14000
9070 RETURN
10000 '*******<<<<<<<<<<BAJAS>>>>>
>>>>>>>>******************
10010 CLS#3:PRINT#3,"
  ( '0' para regresar al menu )"
10020 INPUT#3,"INTRODUZCA POSICION
DEL REGISTRO A BORRAR";nombre:nombr
e$=STR$(nombre):IF i=0 THEN RETURN
ELSE i=0
10030 FOR i=1 TO m
10040 IF nombre$=mat$(i,1) THEN 100
50 ELSE NEXT i:IF i=m THEN RETURN
10050 FOR k=i+1 TO m
10060 FOR j=1 TO jota
10070 mat$(k-1,j)=mat$(k,j)
10080 NEXT i
10090 NEXT k
10100 RETURN
11000 '*******<<<<<<<< modificacion
 ventana ayuda >>>>>>>>***********
****
11010 CLS#3:PEN#3,0:INPUT#3, "INTRO
DUZCA POSICION DEL REGISTRO A MODIF
ICAR";i:CLS#3
11020 PEN#3,0:INPUT#3,"INTRODUZCA N
UMERO DEL CAMPO A MODIFICAR ";C
11030 CLS#3:PRINT#3,"  ";mat$(i,C):
LOCATE#3,10,3:PRINT#3,"  (introduzc
a '#' para regresar al menu) "
11040 LOCATE #3,1,1:INPUT#3,m$
11050 IF m$="#" THEN RETURN ELSE m$
=UPPER$(m$):mat$(i,C)=m$
11060 CLS#1:i=i-1
11070 RETURN
12000 '*******<<<<<<<<<pagina anter
ior>>>>>>>*****************
12005 IF prog<>1 THEN nu=20 ELSE nu
=40:'determina cuantos registros re
trocede
12010 i=i-nu:IF i<1 THEN i=1:CLS#1:
RETURN ELSE CLS#1:RETURN
13000 '*******<<<<<<<<<nuevos regi
stros>>>>******************
13010 CLS#3:PRINT#3,"
('0' para regresar al menu)"
13020 INPUT#3,"INTRODUZCA LA POSICI
ON DEL NUEVO REGISTRO ",posic:IF po
sic=0 THEN RETURN
13025 PRINT#3:INPUT#3,"DESEA UTILIZ
AR EL TECLADO O DISCO (t/d) ",disc$
:disc$=UPPER$(disc$):IF disc$="D" T
HEN REC=0:MODE 2:GOSUB 3030:vent=1:
RETURN
13030 GOSUB 2030:RETURN:'<<<<<retor
no menu >>>>>>>
14000 '***************** continuar
********************
14010 CLS#1:RETURN
14020 RETURN
15000 '*********** fichero ********
*****
15010 LOCATE 2,1:PRINT"POS":LOCATE
8,1:PRINT"TITULO":LOCATE 23,1:PRINT
"TIPO":LOCATE 30,1:PRINT"CINTA":LOC
ATE 37,1:PRINT"VALORACION":LOCATE 5
0,1:PRINT"REVISTA":LOCATE 65,1:PRIN
T"OBSERVACIONES"
15020 LOCATE 2,2:PRINT"---":LOCATE
8,2:PRINT"------":LOCATE 23,2:PRINT
"----":LOCATE 30,2:PRINT"-----":LOC
ATE 37,2:PRINT"----------":LOCATE 5
0,2:PRINT"-------":LOCATE 65,2:PRIN
T"-------------"
15030 WINDOW#0,1,80,23,25:WINDOW#1,
1,4,3,22:WINDOW#2,6,20,3,22:WINDOW#
3,22,28,3,22:WINDOW#4,30,35,3,22:WI
NDOW#5,37,48,3,22:WINDOW#6,50,57,3,
22:WINDOW#7,59,80,3,22
15040 RETURN
16000 '**********
16010 MODE 2:x=0:s=1:WINDOW#1,1,80,
2,21:WINDOW#0,1,80,1,1:WINDOW#3,1,8
0,23,25
16020 IF p=8 THEN PRINT#8:PRINT#8:P
RINT#8,"COD   TITULO         TIPO  C
INTA  VALORACION   REVISTA        O
BSERVACIONES":RETURN
16030 CALL &BB9C:LOCATE 2,1:PRINT"P
OS":LOCATE 6,1:PRINT"TITULO":LOCATE
 21,1:PRINT"TIPO":LOCATE 28,1:PRINT
"CINTA":LOCATE 35,1:PRINT"VALORACIO
N":LOCATE 48,1:PRINT"REVISTA":LOCAT
E 63,1:PRINT"OBSERVACIONES":CALL &B
B9C
16040 RETURN
16045 '***********
16050 PRINT:PRINT"                1.
POSICION":PRINT:PRINT"
  2.TITULO":PRINT:PRINT"
    3.TIPO":PRINT:PRINT"
    4.No. de CINTA":PRINT:PRINT"
            5.VALORACION":PRINT:PRI
NT"             6.REVISTA"
16060 PRINT:PRINT"                7.
OBSERVACIONES"
16070 RETURN
17000 '***********
17010 PRINT#p, USING "\    \";mat$(i
,1);:PRINT#p,USING "\          \"
;mat$(i,2);:PRINT#p, USING "\
\";mat$(i,3);:PRINT#p,USING "\
\";mat$(i,4);:PRINT#p, USING"\
      \";mat$(i,5);
17020 PRINT#p,USING "\        \";ma
t$(i,6);:PRINT#p,USING "\
     \";mat$(i,7)
17030 x=x+1
17040 RETURN
18000 '********************** DIETA
RIO *************************
18010 LOCATE 2,1:PRINT"POS":LOCATE
17,1:PRINT"NOMBRE":LOCATE 34,1:PRIN
T"FECHA":LOCATE 42,1:PRINT"IMPORTE"
:LOCATE 53,1:PRINT"No.HISTO":LOCATE
 67,1:PRINT"OBSERVACIONES"
18020 LOCATE 2,2:PRINT"---":LOCATE
17,2:PRINT"------":LOCATE 34,2:PRIN
T"-----":LOCATE 42,2:PRINT"-------"
:LOCATE 53,2:PRINT"--------":LOCATE
 67,2:PRINT"-------------"
18030 WINDOW#0,1,80,23,25:WINDOW#1,
1,4,3,22:WINDOW#2,6,30,3,22:WINDOW#
3,32,40,3,22:WINDOW#4,42,51,3,22:WI
NDOW#5,53,61,3,22:WINDOW#6,63,80,3,
22
18040 RETURN
19000 '*********
19010 MODE 2:x=0:s=1:WINDOW#1,1,80,
```



```
2,21:WINDOW#0,1,80,1,1:WINDOW#3,1,8
0,23,25
19020 IF p=8 THEN PRINT#8:PRINT#8:P
RINT#8,"POS                NOMBRE
      FECHA    IMPORTE   No.HISTORIA
 OBSERVACIONES":RETURN
19030 CALL &BB9C:LOCATE 2,1:PRINT"P
OS":LOCATE 17,1:PRINT"NOMBRE":LOCAT
E 34,1:PRINT"FECHA":LOCATE 42,1:PRI
NT"IMPORTE":LOCATE 53.1:PRINT"No.HI
STO":LOCATE 67,1:PRINT"OBSERVACIONE
S":CALL &BB9C
19040 RETURN
20000 '***********
20010 PRINT:PRINT"                1.
POSICION":PRINT:PRINT"
   2.NOMBRE":PRINT:PRINT"
      3.FECHA":PRINT:PRINT"
        4.IMPORTE":PRINT:PRINT"
           5.No.HISTORIA":PRINT
20020 PRINT"                 6.OBSERV
ACIONES"
20030 RETURN
21000 '********
21010 PRINT#p, USING "\    \";mat$(i
,1);:PRINT#p,USING "\
          \";mat$(i,2);:PRINT#p, US
ING "\         \";mat$(i,3);:PRINT#p,
USING"####,### ";VAL(mat$(i,4));:PR
INT#p,USING" ####,###   ";VAL(mat$(i
,5));
21020 PRINT#p,USING "\
          \";mat$(i,6)
21030 x=x+1
21040 RETURN
22000 '**************** RECIBOS   **
******************
22010 LOCATE 2,1:PRINT"POS":LOCATE
17,1:PRINT"CONCEPTO":LOCATE 34,1:PR
INT"FECHA":LOCATE 42,1:PRINT"IMPORT
E":LOCATE 67,1:PRINT"PROCEDENCIA"
22020 LOCATE 2,2:PRINT"---":LOCATE
17,2:PRINT"---------":LOCATE 34,2:PR
INT"-----":LOCATE 42,2:PRINT"------
-":LOCATE 67,2:PRINT"-----------"
22030 WINDOW#0,1,80,23,25:WINDOW#1,
1,4,3,22:WINDOW#2,6,30,3,22:WINDOW#
3,32,40,3,22:WINDOW#4,42,51,3,22:WI
NDOW#5,53,61,3,22:WINDOW#6,63,80,3,
22
22040 RETURN
23000 '*********
23010 MODE 2:x=0:s=1:WINDOW#1,1,80,
2,21:WINDOW#0,1,80,1,1:WINDOW#3,1,8
0,23,25
23020 IF p=8 THEN PRINT#8:PRINT#8:P
RINT#8,"POS                CONCEPTO
       FECHA    IMPORTE    SALDO
PROCEDENCIA":RETURN
23030 CALL &BB9C:LOCATE 2,1:PRINT"P
OS":LOCATE 17,1:PRINT"CONCEPTO":LOC
ATE 34,1:PRINT"FECHA":LOCATE 42,1:P
RINT"IMPORTE":LOCATE 54,1:PRINT"SAL
DO":LOCATE 67,1:PRINT"PROCEDENCIA":
CALL &BB9C
23040 RETURN
24000 '*********
24010 PRINT:PRINT"                1.
POSICION":PRINT:PRINT"
   2.CONCEPTO":PRINT:PRINT"
      3.FECHA":PRINT:PRINT"
        4.IMPORTE":PRINT:PRINT"
           5.SALDO":PRINT
24020 PRINT"                 6.PROCED
ENCIA"
24030 RETURN
25000 '*********
25010 PRINT#p, USING "\    \";mat$(i
,1);:PRINT#p,USING "\
          \";mat$(i,2);:PRINT#p, US
ING "\         \";mat$(i,3);:PRINT#p,
USING "####,### ";VAL(mat$(i,4));:P
RINT#p, USING"#####,### ";VAL(mat$(
i,5));
25020 PRINT#p,USING "\
          \";mat$(i,6)
25030 RETURN
26000 '************ descripcion del
 fichero multiple ****************
26010 PRINT"
   FICHERO MULTIPLE"
26020 PRINT"                      ===
=============="
26030 PRINT:PRINT
26040 PRINT"          EL CAMPO POSIC
ION SE RELLENA AUTOMATICAMENTE"
26050 PRINT:PRINT"           EL CAMPO
 TITULO DISPONE DE 14 CARACTERES."
26060 PRINT:PRINT"           EL TIPO
DISPONE DE 6 CARACTERES."
26070 PRINT:PRINT"           EL CAMPO
 CINTA REFLEJA EL NUMERO DE ESTA Y
CONTIENE 5 CARACTERES."
26080 PRINT:PRINT"           EL CAMPO
 VALORACION 10 CARACTERES."
26090 PRINT:PRINT"           EL CAMPO
 OBSERVACIONES 20 CARACTERES."
26100 PRINT"-----------------------
------------------------------------
---------------------"
26110 PRINT:PRINT"NOTA:si desea cam
biar algun campo o modificar su dim
ension modifique las lineas"
26120 PRINT:PRINT"     19000-20000.
"
26130 LOCATE 18,24:PRINT"pulse una
tecla para continuar":CALL &BB18
26140 RETURN
27000 '************ descripcion del
 dietario ****************
27010 PRINT"
 DIETARIO"
27020 PRINT"
 ========"
27030 PRINT
27040 PRINT"           EL CAMPO POSIC
ION SE RELLENA AUTOMATICAMENTE"
27050 PRINT:PRINT"           EL CAMPO
 NOMBRE DISPONE DE 24 CARACTERES."
27060 PRINT:PRINT"           EL CAMPO
 FECHA DISPONE DE 8 CARACTERES."
27070 PRINT:PRINT"           EL CAMPO
 No.HISTORIA CONTIENE 8 CARACTERES.
"
27080 PRINT:PRINT"           EL CAMPO
 IMPORTE 7 CARACTERES."
27090 PRINT:PRINT"           EL CAMPO
 OBSERVACIONES 17 CARACTERES."
27100 PRINT"-----------------------
------------------------------------
---------------------"
27110 PRINT:PRINT"NOTA:si desea cam
biar algun campo o modificar su dim
ension modifique las lineas"
27120 PRINT"     20000-21000."
27130 PRINT:PRINT"Para mayor efecti
vidad del programa introduzca la fe
cha de la forma MM/DD/AA       dond
e MM es el mes con dos digitos, DD
el dia y AA el a/o.
   De esta forma podra listar por m
eses completos."
27140 CALL &BB9C:LOCATE 18,25:PRINT
"pulse una tecla para continuar":CA
LL &BB18:CALL &BB9C
27150 RETURN
28000 '************ descripcion del
 recibos *****************
28010 PRINT"                        C
ONTABILIDAD"
28020 PRINT"                        =
==========="
28030 PRINT:PRINT
28040 PRINT"           EL CAMPO POSIC
ION SE RELLENA AUTOMATICAMENTE"
28050 PRINT:PRINT"           EL CAMPO
 CONCEPTO DISPONE DE 24 CARACTERES.
"
28060 PRINT:PRINT"           EL CAMPO
 FECHA DISPONE DE 8 CARACTERES."
28070 PRINT:PRINT"           EL CAMPO
 IMPORTE CONTIENE 7 CARACTERES."
28080 PRINT:PRINT"           EL CAMPO
 SALDO 8 CARACTERES."
28090 PRINT:PRINT"           EL CAMPO
 PROCEDENCIA 17 CARACTERES."
28100 PRINT"-----------------------
------------------------------------
---------------------"
28110 PRINT:PRINT"NOTA:si desea cam
biar algun campo o modificar su dim
ension modifique las lineas"
28120 PRINT"     21000-22000."
28130 PRINT:PRINT"Para mayor efecti
vidad del programa introduzca la fe
cha de la forma MM/DD/AA       dond
e MM es el mes con dos digitos, DD
el dia y AA el a/o.
   De esta forma podra listar por m
eses completos."
28140 LOCATE 18,25:PRINT"pulse una
tecla":CALL &BB18
28150 RETURN
```

SI BUSCAS LO MEJOR 

Software LO TIENE

OLVIDA TODO LO QUE HAS VISTO

Nº 1 en U.S.A.

U.S. GOLD AI American Software

DATA EAST

KUNG-FU MASTER

THE OFFICIAL COIN OP GAME

LA VERSION OFICIAL DE LAS MAQUINAS

KUNG-FU MASTER

Commodore 64/128

KUNG-FU MASTER
¡¡EL DEFINITIVO¡¡

DISTIBUIDOR EXCLUSIVO PARA ESPAÑA ERBE SOFTWARE C/. STA. ENGRACIA, 17. 28010 MADRID, TFNO.: (91) 447 34 10
DELEGACION BARCELONA, AVDA. MISTRAL, N.º 10 - TFNO.: (93) 432 07 31

GRAFPAD II

¿Quién dice que el ordenador es una fría máquina de cuadricular mentes y axfisiar la creatividad? Esto nos preguntamos después de haber visto funcionar esta portentosa tableta gráfica. La respuesta es, sin duda, que la persona que mantiene esta afirmación, aún no ha dibujado o simplemente no ha visto dibujar con ella.



La conexión es al bus de expansión, y como siempre hay que guardar las clásicas precauciones de conectarla con el ordenador apagado.

Vamos a dibujar

Primero deberemos cargar el programa, del que hay versión en cinta y disco, con RUN **«AINTRO».** Después ya podemos empezar a dibujar.

Pero si queremos hacerlo bien, debemos leer con detenimiento, la completa guía del usuario que acompaña la tableta, ésta afortunadamente, salvo en unos pocos puntos, es bastante clara y nos enseña el funcionamiento de cada icono.

Las opciones de las que disponemos son las siguientes:

— DIBUJO LIBRE: En suma es como dibujar con un simple lapicero. Podremos seleccionar el grosor de la línea.

— SPRAY: Se puede escoger tanto el tipo de spray, como su densidad (9 en total).

— LINEA RELLENA: Nos rellena una línea dibujada desde un punto dado el grosor escogido.

— TRIANGULO: Podemos dibujar todo tipo de triángulos siguiendo los siguientes pasos:

1. Determinar un primer punto con el lápiz en la tableta (primer vértice);

2. Desde este punto, escoger la longitud adecuada del primer lado (el último punto de este lado formará el segundo vértice);

3. Colocar el último punto donde se quiera (tercer vértice).

— CIRCULO: Sólo tenemos que elegir con un punto el centro del círculo y a partir de éste, moviendo el lápiz sobre la tableta determinar la longitud del radio. El círculo saldrá dibujado a continuación.

— ELIPSE: Determinamos el centro de la elipse y moviendo el lápiz en la tableta agrandamos o empequeñecemos la misma, posteriormente saldrá dibujada.

— DIBUJO PUNTEADO: Esta opción se puede escoger después de la del círculo y éste se nos dibujará punteado.

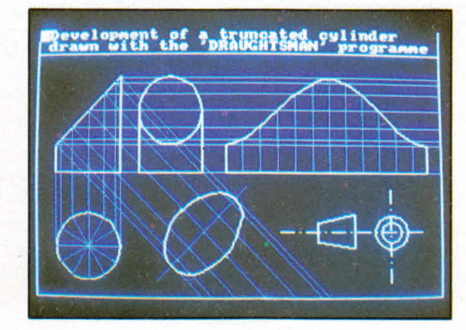
— CUADRADO: Hay que colocar el cursor para elegir el primer punto y mover el lápiz sobre la tableta para definir tamaño y posición.

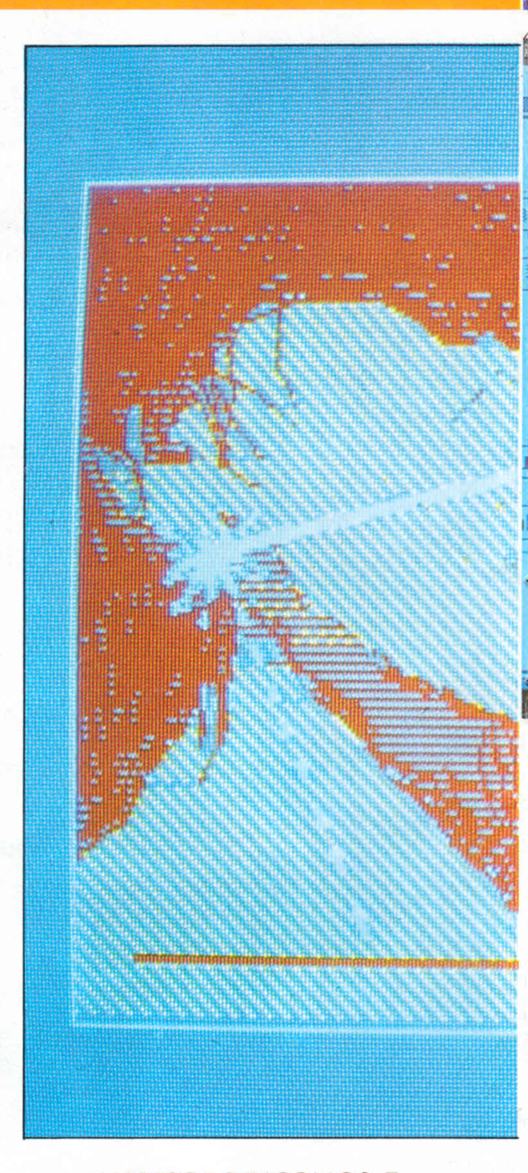
— RECTANGULO: Se siguen los mismos pasos que en la opción de CUADRADO.

— POLIGONOS: Podemos elegir de 3 a 15 lados. Tenemos que marcar un punto en la tableta y apartir de éste elegir el radio del polígono. Después de esto podemos seleccionar el color del mismo.

— GOMA: ¿Nos hemos equivocado? Pues no importa, para eso tenemos borrador, y de éste podremos elegir el color de la tinta a borrar, borrando sólo ésta. Podremos elegir también el tamaño de la goma.

Después, como si de un típico borrador se tratara, moveremos en la tableta el lápiz para borrar la superficie deseada.





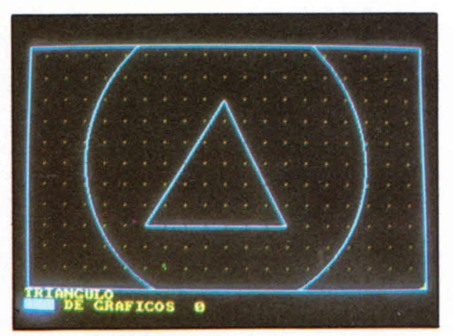
— IMPRESOR DE ICONOS: Tenemos con esta opción la posibilidad de acceder a un archivo de almacenaje de 32 iconos, este archivo está compuesto de dos páginas de 16 iconos, y para pasar de una página a otra deberemos situar el cursor sobre el último, realzado en rojo, que sirve para este fin.

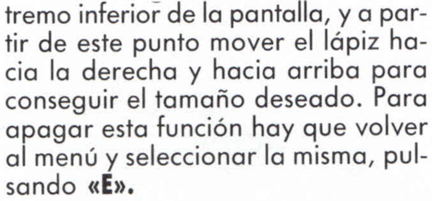
Una vez elegido el icono debemos poner el lápiz sobre el pulsor **«E»,** posteriormente lo situaremos en la zona de la tableta que deseemos, pulsaremos **«E»,** de nuevo y el icono quedará en la zona de pantalla definida con la tableta. Podemos usar nuestros propios iconos si previamente los hemos creado con el GENERADOR DE ICONO, una vez hecho esto los podemos salvar con la opción GRABAR Y CARGAR, luego, con la misma opción también, podemos cargarlos en el impresor y utilizarlos.

— VENTANA DE GRAFICOS: Nos permite difinir la porción de pantalla en la que queramos dibujar.

Debemos fijar un punto en el ex-






tremo inferior de la pantalla, y a partir de este punto mover el lápiz hacia la derecha y hacia arriba para conseguir el tamaño deseado. Para apagar esta función hay que volver al menú y seleccionar la misma, pulsando **«E»**.

— BORRADOR DE PANTALLA: Esta opción permite limpiar totalmente la pantalla. Hay que seleccionar el icono y después confirmar con las opciones **«YES»** o **«NO»** el borrado de la pantalla.

— CURSOR COMPLETO: Con el cursor completo tenemos la posibilidad de utilizar un centrador de pantalla, para tomar referencias, etc.

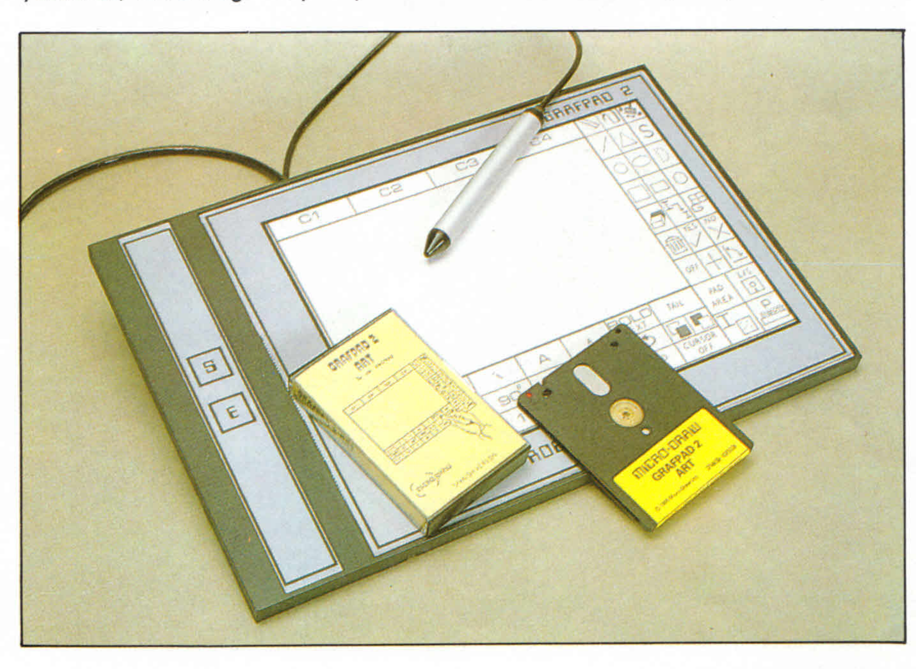
— LINEAS HORIZONTALES Y VERTICALES: Debemos seleccionar el grosor de la línea, fijar un punto pulsando **«E»**, y dibujar hasta donde queramos, volver a pulsar **«E»** y tendremos la primera línea desde la cual ya podremos dibujar todas las horizontales y verticales que queramos pulsando **«E»** cada vez que se quiera marcar.

— TEXTO: En esta opción podemos usar letras normales mayúsculas y minúsculas, itálicas (inclinadas) hacia adelante y atrás, también en mayúsculas, letra negrilla y la posibilidad más vistosa que es la proyección del texto, pudiendo cambiar el color de la proyección referente al color del texto.

— LLENADO: Podemos rellenar tanto la pantalla completa, como cualquier superficie que hallamos delimitado previamente.

Si queremos que sea de un solo color debemos seleccionar el mismo color de primer plano y el mismo color de fondo. Una vez hecho esto llevaremos el cursor dentro de la superficie a rellenar y pulsaremos **«E»,** el relleno se hará de acuerdo con la altura en la que hallamos situado el cursor, esto es, que si se pone a la mitad, por ejemplo, sólo llenará la mitad de la superficie.

Pero también podemos utilizar las 32 muestras que tiene almacenados el programa, muestras en las que también deberemos escoger los colores de fondo y de primer plano.



— EJE: Esta opción está destinada al giro de la forma creada y podemos girarla 90, 180 y 270 grados. Después la archivaremos poniendo el cursor dentro y pulsando **«E».**

Si deseamos cambiar la muestra, el color, o ambos a la vez, como si queremos hacer un nuevo giro, esto lo lograremos parando el archivo con ESCAPE.

— GRAFICOS EN COLOR (GCOL): Se debe elegir 1, 2 ó 3 y utilizar cualquier función de dibujo para variar los efectos de la muestra.

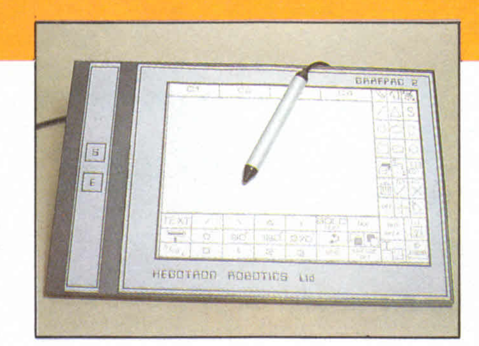
— REJILLA (GRID): Al elegir esta opción se nos pedirá en pantalla que determinemos el tamaño de la rejilla, mediante la separación, por pixel, de sus líneas.

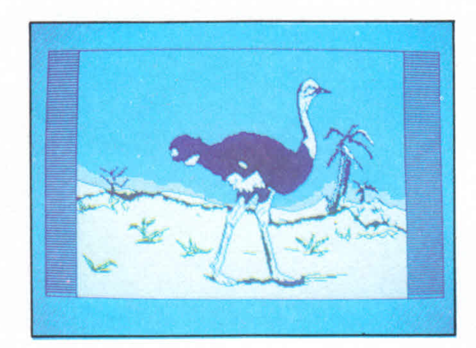
— CURSOR OFF: Hace desaparecer el cursor de la pantalla.

— AREA DE LA TABLETA (PAD AREA): Podremos delimitar el área a utilizar de la pantalla.

— GRABAR Y CARGAR: Con esta opción podemos salvar y cargar nuestras creaciones.

— GENERADOR DE ICONOS: Para la versión en cinta del programa, deberemos reinicializar el ordenador (CTRL + SHIFF + ESC) y cargar la segunda cara de la cinta.





En la versión de disco se apuntará al icono del menú de la tableta, se nos pedirá conformidad para cargarlo, (S/N), y si la damos cargaremos el generador, saliendo del programa principal.

Si bien el acceso al generador desde el programa principal es rápido y simple, el acceso a éste desde el generador es laborioso, exigiendo la grabación de nuestro diseño, reinicialización del ordenador y posterior carga, otra vez, del programa principal.

Pero volviendo al funcionamiento del generador, podemos usar los 32 caracteres residentes o diseñar los nuestros mediante el siguiente sistema:

Colocaremos el cursor en la rejilla de diseño y moveremos el lápiz sobre la tableta para mover el cursor, cuando queramos imprimir o borrar pulsaremos **«E»** (teniendo en cuenta que el rojo imprime y que el blanco borra).

Una vez diseñado el carácter, colocaremos el cursor en la rejilla mostradora de iconos y pulsaremos **«E».**

Para almacenar, arrastraremos el carácter al archivo de almacenaje y pulsaremos **«E».** De la parrilla de diseño, siempre con el carácter creado, pasaremos a la rejilla de impresión para hacer caracteres mayores. Ellos tienen que ser almacenados como caracteres individuales en el archivo de almacenaje.

El ganador de iconos tiene las siguientes opciones:

— Puede invertir el carácter en la rejilla de diseño.

Section of an ordnance survey map showing another use of symbol function.



— Dar una imagen espejo del carácter referente al eje X o al eje Y.

— Girar a 90, 180 y 270 grados el carácter o icono generado.

— Limpiar, según queramos, la rejilla de impresión, o la rejilla diseñadora.

— Grabar o cargar los caracteres generados.

— Rutina de impresora si queremos imprimir la imagen en pantalla. La rutina es para las EPSON MX o FX.

Una vez explicadas las amplísimas posibilidades de GRAFPAD II, queda por decir que todas sus opciones se escogen pulsando la **«E»** que está en la tableta y que es equivalente a ENTER, para abandonarlas hay que poner el lápiz sobre la casilla de **«OFF»** y pulsar **«E».** Pero según datos facilitados por el distribuidor, el próximo modelo de GRAFPAD II irá dotado de un interruptor en el lápiz, desapareciendo la casilla **«E»** y simplificando su utilización con una sola mano.

¿Y cómo dibuja GRAFPAD II?

Bien, francamente bien, y pasaremos muy buenos ratos dando rienda suelta a nuestra imaginación haciendo todo tipo de dibujos e **«inventos»** gráficos llenos de color y vivacidad.

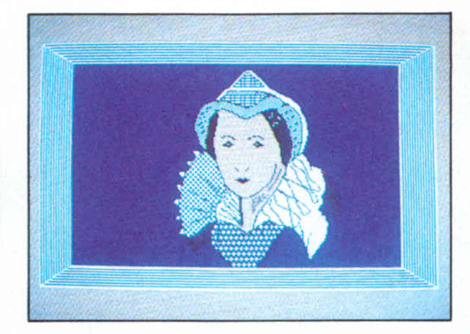
Pero como era lógico no nos conformamos con nuestra modesta opinión de aficionados al dibujo, y pedimos a un profesional, Manuel Barco, nuestro portadista, su parecer sobre la tableta gráfica y sus conclusiones han sido éstas:

El GRAFPAD II ofrece muchos aspectos positivos principalmente a diseñadores gráficos para la ejecución de bocetos e ideas.

En lo referente a las artes finales no es comparable a la ilustración o diseño sobre papel, aunque tal vez lo que no lo hace comparable sea el tamaño de la retícula de la pantalla, que al ser muy grande hace que sean bastas e imprecisas las líneas y las medias tintas.

Con el color se podrían haber ofrecido mayor cantidad de matizaciones si las opciones de la gama de colores pudieran corresponderse con los tres colores primariso ópticos (Magenta, Cian, Amarillo).

Una de las posibilidades que más sobresalen es el poder cambiar de color sobre el trabajo ya realizado.



De esta manera se puede ofrecer, sobre una misma idea, muchas variaciones de color.

Asimismo y en relación con las medias tintas, es también destacable la opción de **«llenado»** y su variedad de textura. Aislando las formas puede variarse el color del punto de la trama y ofrecer distintas tonalidades dentro de una misma gama cromática. Aquí es necesario señalar la imprecisión de la opción **«Spray»** en este menester, aunque en otros sea práctico, como por ejemplo, para usarlo en el momento en que el límite entre dos texturas sea muy violenta, y en ese caso una pasada de spray ayuda a romper el límite cortante.

Como de muy práctica se puede definir la opción **«Línea rellena»,** no sólo para hacer líneas de gran longitud o de construir líneas geométricas, sino que es útil también para delimitar letras o formas muy precisas, en las que el límite tiene que ser nítido y no necesariamente recto, ya que a trazos cortitos podemos conseguir con mayor seguridad que con **«Dibujo libre»** un trazado satisfactorio.

Y en el área del dibujo técnico es de agradecer la retícula de la que podremos disponer cada 2, 4, 8, 16 y 32 pixel según optemos. Con esta retícula se pueden situar centros de circunferencias, puntos de fuga, mediciones, etc. Esta retícula se puede hacer desaparecer sin que transcienda al trabajo.

Pues después de haber podido contar con esta acreditada opción, poco más tenemos que decir de esta fabulosa herramienta gráfica, que hará nuestras delicias, si aprendemos a manejarla con destreza.

GRAFPAD II FICHA TECNICA

Fabricante: **Micro Draw**

Distribuye: **Ofites Informática.**

Contenido del Kit: **Tableta gráfica, Software en cinta o disco según versión.**

Sistema operativo: **Amsdos.**

Precio: **23.900** cassette, **25.900** disco. (IVA no incluido).

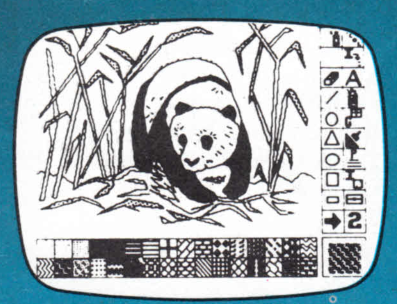
Ofites Informática

Presenta: la tableta gráfica

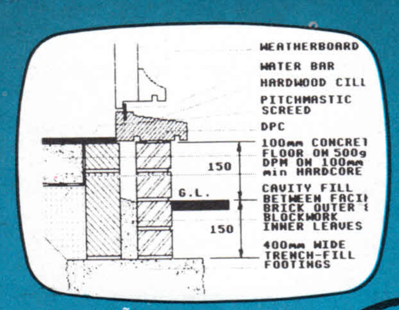
GRAFPAD II-

LO ULTIMO EN DISPOSITIVOS DE ENTRADA DE GRAFICOS PARA AMSTRAD, COMMODORE Y BBC

La primera tableta gráfica, de bajo costo, en ofrecer la duración y prestaciones requeridas por las aplicaciones de negocios, industria, hogar y educación. Es pequeña, exacta y segura. No necesita ajustes ni mantenimiento preventivo. GRAFPAD II es un producto único que pone la potencia de la tecnología moderna bajo el control del usuario.



DIBUJO A MANO ALZADA
SOFTWARE DE ICONOS



DISEÑO DE ARQUITECTURA
CON SOFTWARE DDX

ESPECIFICACIONES

RESOLUCION:
1.280 x 1.024 pixels.

PRECISION:
1 pixel.

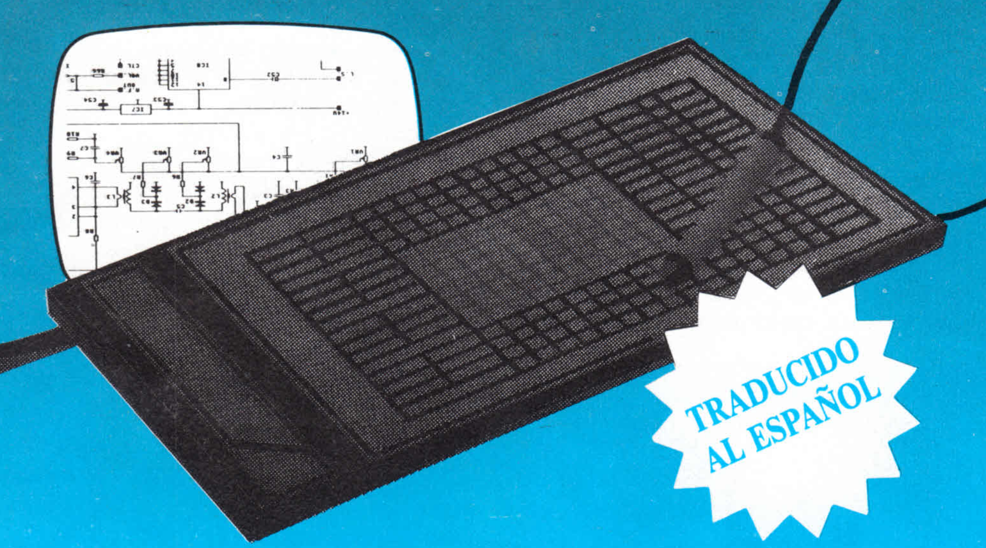
TASA DE SALIDA:
2.000 pares de coordenadas por segundo.

INTERFACE:
paralelo.

ORIGEN:
borde superior izquierdo o seleccionable.

DIMENSIONES:
350 x 260 x 12 mm.

DISPONIBLE AMSTRAD:
CASSETTE .... 23.900 ptas.
DISCO .... 25.900 ptas.
(IVA NO INCLUIDO)



- FACIL DE USAR.
- TRAZADO PCB.
- C.A.D.
- AREA DE DISEÑO DIN A4.
- COLOR EN ALTA RESOLUCION.
- USO EN HOGAR Y NEGOCIOS.
- VARIEDAD DE PROGRAMAS DISPONIBLES.
- DIBUJO A MANO ALZADA.
- DIAGRAMAS DE CIRCUITOS.

COMBINA EN UN UNICO DISPOSITIVO TODAS LAS PRESTACIONES DE LOS INTENTOS PREVIOS DE MECANISMOS DE ENTRADA DE GRAFICOS. LAS APLICACIONES SON MAS NUMEROSAS QUE EN LOS DEMAS DISPOSITIVOS COMUNES E INCLUYEN:

• selección de opciones • entrada de modelos • recogida de datos • diseño lógico • diseño de circuitos • creación de imágenes • almacenamiento de imágenes • recuperación de imágenes • diseño para construcción • C.A.D. (diseño asistido por ordenador) • ilustración de textos • juegos • diseño de muestras • educación • diseño PCB.

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener la tableta gráfica, puede dirigirse a:



Avda. Isabel II, 16 -8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES



Tasman SOFTWARE

por fin en España, software a precios británicos

TASWORD

¿Se imagina su ordenador convertido en una máquina de escribir? TASWORD es la mejor relación calidad-precio en tratamiento de texto profesional.

Totalmente en castellano, permitiendo realizar MAIL MERGE, trabajar en bloques sin ninguna interrupción incrementando su velocidad, etc... (en versión 6128 aprovecha las 128 K creando un disco virtual de 64 K).

- Acentos, ñ, ü, ¿, etc...
- Compatible Productos TASMAN.
- Adaptación impresoras.
- Configuración propia por usuario.
- Ensamblaje de textos.



9.900 pts. AMSTRAD COMMODORE EINSTEIN MSX

6.900 pts. AMSTRAD COMMODORE MSX SPECTRUM

7.900 pts. SPECTRUM

TAS-SPELL

Primer auxiliar que corregirá la ortografía de sus escritos y pondrá los acentos olvidados no dando margen a ningún error.

Contiene un potente diccionario con más de 20.000 vocablos pudiendo Vd. ampliarlos. Complemento ideal para su TASWORD con disco.



7.600 pts. AMSTRAD

Próximamente en versión PCW 8256 8512

TAS-PRINT

Con TAS-PRINT la escritura elevada a arte. Utiliza las grandes posibilidades gráficas de su ordenador. Las posibilidades tipográficas las explota al máximo al dar una doble pasada optimizando la calidad.

Los tipos de escritura son: COMPACTA MEDIAN DATA-RUN LECTURA LIGHT PAICE SCRIPT



7.600 pts. AMSTRAD EINSTEIN

5.900 pts. AMSTRAD SPECTRUM

6.900 pts. QL SPECTRUM

TASCOPY

Sin necesidad de un PLOTTER podrá obtener sus gráficos de pantalla a través de la impresora.

Un increíble ZOOM le permite realizar sus gráficos en 4 hojas formando un póster de gran tamaño.

7.600 pts. AMSTRAD

5.900 pts. AMSTRAD SPECTRUM

6.900 pts. QL SPECTRUM

GRAFMAN

Programa de E.G. Computer Graphics especialmente diseñado para trabajar conjuntamente con TASCOPY representando las funciones matemáticas en desarrollos de diagramas por coordenadas, permitiendo su efecto "ZOOM" ampliar sectores de dichos diagramas.



5.600 pts. 6.200 pts. SOLO AMSTRAD



- IVA NO INCLUIDO

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:

Ofites Informática

Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA Y PORTUGAL

DOMINAR EL CPM ES TAREA FACIL

Francisco G. R.

Hasta hace poco tiempo la mayoría de los usuarios de microordenadores se planteaban la dificultad del Basic. Una vez resuelto el problema se intentaba aprender un poco más y se planteaban los problemas del código máquina. Ultimamente y con motivo de la aparición en el mercado del Amstrad, un nuevo término les empezó a preocupar: «Sistema Opertativo CP/M»



Muchos se preguntarán qué es el CP/M, para qué sirve, o similares preguntas. Otros se repetirán que es un Sistema Operativo. Pues bien, vamos a intentar explicar qué son ambas cosas.

Sistema Operativo y CP/M

El Sistema Operativo es el cerebro del microordenador. Este se configura para tener **«conocimientos»**, de los periféricos que se le pueden concectar entre sí, ahora o en un futuro. El sistema operativo controla todos los equipos, pero normalmente, al igual que el CP/M, no realiza ningún procedimiento complicado.

El CP/M es un Sistema Operativo en Disco (DOS: Disc Operating CP/M son las siglas de Control Program Monitor, y existen en el mercado varias versiones del mismo. Entre las más importantes están el CP/M 80 y el CP/M 86. La primera se usa en casi todas las máquinas cuya CPU sea un 8080 o Z80, esto es, un microprocesador de 8 Bits, con unidad de disco. La segunda funciona con las máquinas que lleven un 8086 u 8088, al igual que antes, nos referimos a las máquinas que tengan un microprocesador de 16 Bits, y su respectiva unidad de disco.

La Historia del CP/M

Retrocediendo unos cuantos años atrás, podemos observar los comienzos y divulgación del CP/M. El Sistema fue desarrollado por el

Dr. Gary Kildall en el 1973. La primera versión se lanzó para el Sistema experimental Kildall, que incluía la primera unidad de disco de 8'' de la casa Shugar Asociates.

Durante varios años fue creciendo el mercado de ordenadores, y los poseedores de éstos no estaban preparados ni seguros de lo que tenían entre las manos. Por entonces las casas desarrollaban, bien paralelamente o posteriormente en sus propios sistemas. Esto daba lugar a retrasos en los proyectos y el coste de los mismos.

Las pequeñas empresas decidieron adoptar el sistema de Kildall y abaratar el coste, olvidándose de la investigación y desarrollo de los sistemas. Esta es una de las causas principales del empujón que se le dio al CP/M, el cual llegó a ser uno de los casi más utilizados del mercado actual.

Pronto los usuarios empezaron a desarrollar aplicaciones para el CP/M, potenciándolo a nivel de intercambios de información de unos ordenadores a otros. La casi compatibilidad entre los equipos que tuvieran un 8080 o Z80, era lo que permitía a estos usuarios y no usuarios el intento de nuevos programas para mayor utilización de éste. Al lanzar rápidamente sus productos, las compañías no podían entregar una amplia información de ellos.

Con lo cual los comercios de ordenadores no podían prestar suficiente servicio técnico al usuario.

Con la incorporación de las unidades de disco, los técnicos se lanzaron a un gran desarrollo de Software, no sólo programas a medida sino programas de utilidades. Entre estos programas de utilidades podemos destacar las primeras versiones del EBASIC y CBASIC.

Más adelante se le incorporaron otras herramientas, lenguajes de programación, que le dio gran impulso y popularidad.

En 1981 con la implantación del sistema en marcas como IBM, Hewlett-Packard y Xerox, hicieron que las ventas del sistema se duplicaran.

Hoy en día, disponemos de versiones más avanzadas de las que realizó Kildall por primera vez. Las versiones se fueron ampliando y mejorando sustancialmente. Se pasó de la versión 1.4 a la 2.0 y después a la 2.2 (la cual se incluye en la compra de nuestro ordenador). Actualmente se está trabajando sobre la 3.0 o PLUS. Esta última, trabaja con 128K, dejando mayor capacidad para el usuario. El CP/M plus se coloca en el segundo banco de memoria, dejando libre el primero para trabajo.

Estructura interna del CP/M

El CP/M como hemos dicho es un sistema operativo; éste tiene una estructura que consta de bloques de programas, comandos, etc. En esta estructura cabe destacar tres partes: CCP, BDOS y BIOS. Tal vez nos sean familiares estos nombres pero posiblemente desconozcamos qué son y para qué sirven.

El CCP (Console Command Processor: procesador de órdenes de consola), es la primera parte que se carga en memoria después de la página base y la TPA (la página base y la TPA, es la zona de transición de programas). Este es el encargado de interpretar las órdenes que se introducen por el tecaldo. El CCP reconoce solamente unas cuantas órdenes y unos caracteres de control que más tarde explicaremos.

El BDOS (Basic Disk Operating System: Sistema Operativo de Disco Básico), se coloca a continuación del CCP. Toda la actividad del disco y de la consola pasa a través de esta parte. El BDOS no es accesible mediante órdenes directas, sino que se necesita de un programa transitorio o del propio CCP. que nos carga un registro de memoria **(el registro C)**, con la orden deseada, y posteriormente se ejecuta una instrucción de CALL a la posición &HOOO5.

El BIOS (Basic Input/Output System Básico de Entrada/Salida), es la última parte de la que consta el CP/M. El BIOS es particular de cada casa. Cada una, lo diseña de acuerdo a sus necesidades, bien por arquitectura del equipo o necesidades de éste para posteriores interconexiones. Antes de intentar conectar otro sistema a nuestra máquina, necesitamos adaptar el BIOS a este nuevo. En el BIOS, encontramos todos los parámetros del controlador de disco, entre otros. Muy ùtiles, si queremos conexionar una unidad de disco de 5''1/4, o bien efectuar una transmisión y recepción de datos mediante un interface RS-232-C.

Códigos de control de Consola del CP/M 2.2 (464 y 6128)

— Control-C: Arranque en caliente

Es la instrucción de arranque en caliente. Se limita a copiar de nuevo en su interior el CCP, con lo que se muestra como si hubiéramos cargado el sistema de nuevo.

— Control-E: Retorno del cursor.

Nos retorna el cursor al final de la línea. Este final hay que decir que no es un final lógico sino más bien físico.

— Control-G: BEL.

Como su nombre indica es el carácter encargado de producir el pitido «Bel». Es el valor de la tabla ASCII número 7, que es el encargado del timbre.

— Control-H: Delete.

Esta orden de consola, está basada en el movimiento del cursor un carácter hacia la izquierda, permitiendo escribir desde allí. Al efectuarnos el movimiento nos escribe un carácter en blanco, con lo que prácticamente es un delete hacia la izquierda del cursor.

— Control-I: Tabulador.

Funciona exactamente igual que la tecla TAB. Nos manda el cursor al siguiente punto de tabulación o bien nos lo mueve unos caracteres más adelante a través de la línea de texto, según esté prefijada.

— Control-J: Salto de línea.

Nos realiza un salto de línea, pasándonos a la anterior para seguir insertando otra línea. Prácticamente es un retroceso de carro, al igual que en una máquina de escribir.

— Control-M: RETURN.

Este comando se comporta igual que si pulsáramos el RETURN o el ENTER. Tiene la misma función que los anteriormente mencionados.

— Control—P: Switch de impresora.

Este comando es uno de los más interesantes que tenemos. Es el Switch de la impresora. Cuando lo pulsamos por primera vez se queda conectado, reflejándose toda la información de la pantalla en la impresora. Cuando se vuelve a pulsar ésta se desactiva. Este comando lo trataremos más adelante con mayor amplitud, al hablar del mismo en el CP/M Plus.

— Control-R: Repetir línea.

Si estamos escribiendo una línea de programa y hemos errado varias veces, la pantalla puede estar no muy clara, con lo cual se utiliza este comando, repitiéndonos la línea tal y como se encuentra en el interior de la memoria.

— Control-S: Bloquear la pantalla.

En la ejecución de un programa, al querer observar algún detalle del mismo, como el funcionamiento o cálculo de éste, lo utilizamos para detener momentáneamente la ejecución de este. Para reanudarlo basta con pulsar cualquier tecla.

— Control—U: Delete total de línea.

Este comando nos borra la línea entera en la cual se encuetre el cursor en ese momento. Tiene una acción de LINE FEED totalmente destructiva.

— Control-X: Delete total hacia la Izquierda.

Un código muy parecido al Control-U, nos borra toda la línea. Con la diferencia que este código solo no borra hacia la izquierda, pero apenas podemos diferenciarlo con respecto al anterior.

Códigos de Control de Consola del CP/M Plus (6128 y 8256).

— Control-A: Cursor izquierda.

Nos mueve el cursor un carácter a la izquierda. Podemos insertar cualquier carácter en donde situemos el cursor. Queda limitado a una misma línea, o sea que cuando lo tenemos al comienzo de línea no podemos pasar a la línea superior.

En el PCW 8256, mover el cursor hacia la izquierda lo efectuamos mediante la tecla de dirección de izquierda.

— Control-B: Cursor a los extremos.

Al pulsarlo por primera vez, y no estando en el principio, nos manda el cursor al principio de la línea. Si se pulsa por segunda vez nos remite el mismo al final de ésta. Al igual que la anterior, este comando se limita a una sola línea no pudiendo ir a otra inferior o superior.

Esta orden varía en el ordenador 8256, podemos diferenciar dos pasos de cursor.

Mediante la función LINEA (se obtiene pulsando a la vez MAYS+FDL), lo moveremos al principio de línea. Para ir al final de línea tendremos que pulsar la misma tecla pero sin MAYS (sólo el FDL).

— Control-C: Arranque en caliente.

Al introducir esta orden el CCP iniciará una operación de arranque en caliente. Esta operación reinicializa el CP/M completamente, incluyendo los valores del controlador de Disco.

El efecto que produce en el interior es una copia del CCP en la memoria, relee el directorio de la unidad por defecto y reconstruye el mapa de situación que contiene el BIOS.

Cuando lo introducimos en medio de una línea, al llegar a él, el ordenador aborta lo anterior, reinicializándose el CP/M.

— Control-E: Retorno del cursor.

Cuando damos la orden Ȼ, el CP/M envía una orden de CARRIAGE RETURN, LINE FEED a la consola, pero no comienza a ejecutar la línea de orden que se ha escrito hasta el momento. La orden es situarse al final de la línea física y no lógica.

— Control-F: Cursor derecha.

Nos manda el cursor hacia la derecha un carácter. Al igual que A, está restringida a una sola línea, no podemos seguir avanzando hacia la derecha, si tuviéramos otra línea. Este movimiento de cursor no es destructivo, no borra el carácter. Esta orden la tenemos en el cursor de la izquierda para el PCW 8256.

— Control-G: CLR.

Funciona como la tecla de función CLR. Nos borra el carácter en el cual está posicionado el cursor. Nos borra el carácter y nos corre la línea un carácter hacia la izquierda. En el PCW lo obtenemos con la tecla de Delete hacia la derecha.

— Control-H: Delete.

Es el carácter de control del ASCII encargado del retroceso. Cuando pulsamos el Ħ, el CCP nos mueve el cursor una posición hacia atrás destructivamente. Se utiliza para corregir los errores de mecanografía al introducir una instrucción. El último carácter que tengamos en la pantalla desaparecerá. El CCP realiza una secuencia de caracteres de retroceso, espacio, retroceso de consola.

Esta orden la tenemos un poco más ágil en el PCW, al igual que el CLR el Delete lo encontramos en Borrar hacia la izquierda.

— Control-I: Tabulador.

Nos manda el cursor hasta el próximo punto de tabulación. Es una forma rápida de movernos a través de la línea de texto. Podemos ir hacia delante dentro de una misma línea, teniendo en cuenta que podemos insertar caracteres en cualquier posición en que quede situado el cursor.

— Control-J: Salto de línea.

Nos da el mismo efecto que si utilizáramos el código ASCII de LINE FEED. El CCP nos ejecuta el salto de línea en la consola. Nos mueve el cursor una posición hacia abajo, y nos lo coloca al principio de línea.

Para el PCW 8256 obtenemos el mismo resultado pulsando ALT+Cursor abajo. Nos lleva el cursor a la siguiente línea sin ejecutárnosla.

— Control-K: CLR total hacia la derecha.

Nos elimina los caracteres desde donde se encuentre el cursor hasta el final de línea. Esta orden se utiliza cuando estemos modificando líneas editadas para su mejor correción. Esta orden también la encontramos en el 8256 pulsando al mismo tiempo ALT y la tecla de Borrar hacia la derecha.

— Control-M: RETURN.

En nuestro ordenador disponemos de dos teclas para la introducción de órdenes por teclado al ordenador, son el RETURN y el ENTER (o INTRO en España). Pues bien, disponemos de otra forma de meter órdenes a través del teclado no tan cómoda pero realiza la misma función, M nos ejecuta un CARRIE-RETURN.

— Control-P: Switch de impresora.

Esta orden funciona como un interruptor de la **«luz».** Si lo pulsamos se activa y si lo volvemos a pulsar se desactiva. Cuando está conectado, todos los caracteres enviados a la consola se dirigen también al mecanismo de listado del CP/M.

Si nos hemos planteado alguna vez cómo poder realizar un listado de un CAT o un DIR de un disco, con este comando lo hemos resuelto.

Si activamos la impresora con P y a continuación le pedimos un DIR, automáticamente nos quedará reflejado en el papel el mensaje del Promt y la orden de DIR, y todo el contenido del disco.

Como anécdotas, podemos decir también que este código lo encontramos en el CPC 6128 en la tecla de CLR. Cuando la pulsamos por primera vez activamos el canal de la impresora y si volvemos a pulsarlo lo desactivamos.

Cuando realicemos un arranque en caliente, el canal de la impresora se encuentra en off. Uno de los defectos que tiene esta utilización es que los controladores internos de la impresora no se comportan de forma muy inteligente si la impresora está desconectada o no preparada cuando el programador o el programa le piden que impriman. Con lo cual debemos tener mucho cuidado con la utilización de este comando.

Pues si no tenemos preparada la impresora y por casualidad nos hemos equivocado al pulsar la tecla de borrar (Delete) un carácter, y por estar tan próximas hemos pulsado las de CLR, el ordenador se nos colgaría y tendríamos que inicializar el equipo por completo. Esta es una orden directa para el PCW 8526, se encuentra en la tecla de función F7/F8.

— Control-Q: Desbloquear la pantalla.

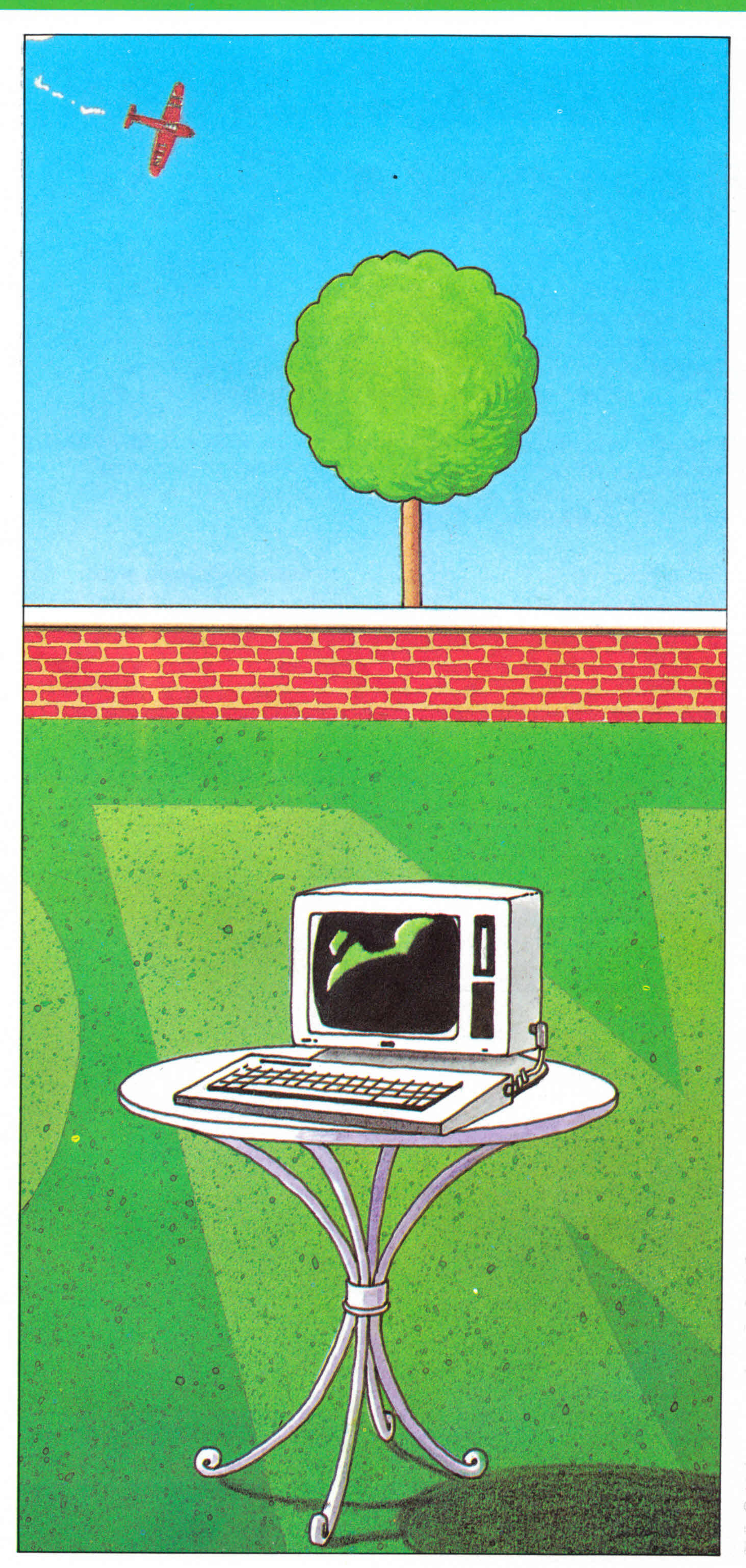
Cuando utilizamos una detención de la pantalla para la comprobación de datos o buena realización del programa, utilizamos esta orden para renanudar su curso. Normalmente se emplea después de introducir un Control-S. En el PCW la obtenemos automáticamente pulsando F3/F4.

— Control-R: Repetir línea.

Cuando ejecutamos una orden Control-R, hacemos que el CCP repita o vuelva a escribir la línea en curso.

El ordenador nos emite el carácter **«#»**, un CARRIAGE RETURN/LINE FEED, y después todo el contenido de la memoria interna de línea de órdenes.

Esta orden también varía un poco en el



PCW, en éste la tenemos en una sola tecla **«JUST»**. Podemos empezar a escribir desde el principio de la línea hasta la situación actual del cursor.

— Control-S: Bloquear la pantalla.

El comando Control-S nos detiene momentáneamente la ejecución **(o salida por consola)** de un programa. El efecto es idéntico al que se produce cuando durante un listado por pantalla pulsamos la tecla de ESC, la pantalla se nos bloquea para poder apreciar su información o la realización de cierto programa. La orden es el carácter de ASCII XOFF, es la abreviatura de **«Transmit Off»**. Podemos decir que es un interruptor para conectar y desconcectar la salida por la consola. Podemos desbloquearlo mediante la pulsación de otra tecla o el propio S. Pero lo más normal es que se realice mediante un Q, evitando de esta manera la posibilidad que se rompa el progocolo de comunicación que exista con la consola. En el 8256 la tenemos directamente en la tecla de función F5/F6.

— Control-U: Delete de línea.

Otro comando muy práctico y peligroso a la vez. Nos elimina la línea en la cual tengamos posicionado el cursor. Este comando normalmente lo llevan todos los procesadores de textos siendo de gran utilidad para la depuración de documentos. Al borrar la línea introducida podemos anular todos los errores cometidos y empezar de nuevo. La orden se nos muestra en pantalla con un carácter «#», después de un CARRIE RETURN/LINE FEED y algunos espacios en blanco para dejar alineado el cursor.

— Control-W: Repite la línea anterior.

En el interior de nuestra memoria, se busca la última línea que se encuentra en el buffer y nos la manda imprimir en la pantalla. Esto se puede realizar siempre y cuando no hayamos introducido ningún carácter en la nueva línea. Si hemos introducido alguno, lo borraríamos y llamaríamos a la anterior línea.

— Control-X: Delete total hacia la izquierda.

Nos realiza un movimiento del cursor hacia el principio de línea de forma destructiva. Nos elimina todos los caracteres que tengamos hacia nuestra izquierda dejándonos los de la derecha intactos. Al igual que K, tenemos la misma facilidad de uso. Ahora utilizaremos la misma función tan sólo variando la tecla de borrado hacia la izquierda.

— Control-Z: Final de cadena.

Esta orden la emplearemos cuando hallamos realizado una cadena con instrucciones como PIP y ED, al final de la misma, para indicar al CCP que es un fin lógico de cadena. Esta orden la obtenemos directamente en el PCW pulsando la tecla de función F1/F2.

COMO PODER LLENAR MAS MEMORIA RAM

Si hojea las páginas de Microhobby Amstrad comprobará que una gran parte de los listados de los programas que habitualmente publicamos son bastante cortos. Teniendo cerca de 40K de RAM libres para llenarlos de instrucciones Basic podemos decir que realmente utilizamos muy poca memoria de nuestro Amstrad cada vez que escribimos y ejecutamos uno de estos programas

¿Qué le parece la idea de poder almacenar varios programas Basic a la vez en la memoria? Y no sólo eso, sino que si además pudiéramos relacionar unos con otros y hacer que se ejecutara cualquiera de ellos... sería una maravilla.

Tendríamos la posibilidad de reunir en la memoria, y al mismo tiempo, todos los programas de la serie de Primeros Pasos, por ejemplo, de una semana. Podríamos seleccionar uno, ejecutarlo, seleccionar otro, ejecutarlo a su vez y comparar los resultados. Y también conseguiríamos evitar teclear repetidamente un listado o tener que acceder al disco o a la cinta con demasiada frecuencia.

Estas pudieran ser algunas de las múltiples aplicaciones que tendríamos al alcance de nuestra mano una vez que podamos disponer de esta facilidad. Nos abre todo un nuevo y extenso campo de posibilidades a la hora de utilizar nuestro ordenador.

Por ejemplo, podríamos emplear un programa para modificar otro. No nos sería nada difícil escribir una utilidad para resumir un programa, eliminando todos los REMs que hubiera en su listado. Otra cosa que se nos ocurre es hacer una de búsqueda y sustitución que nos permita cambiar los nombres de las variables, etc.

Hemos citado algunas ideas que se nos han ocurrido, pero seguro que en este momento estarán pasando por su mente muchas más.

El Programa que acompaña este artículo nos permitirá almacenar y ejecutar programas Basic en cualquier dirección de la memoria. Esto quiere decir que vamos a poder cargar un programa en la posición &1000, por ejemplo, otro en la &2000 y un tercero en la &3000. Y además nos será posible seleccionar uno de ellos y ejecutarle.

Tenemos la facilidad de elegir cualquier cifra como dirección de comienzo de carga pero precisamente por eso le aconsejamos que no lo haga alegremente. Escoja unas cantidades semejantes a &1000, que sean faciles, ya que si así lo hacemos va a resultarnos relativamente simple mantenernos siempre al tanto de donde hemos colocado cada uno de nuestros programas.

Es necesario, también, que nos aseguremos de que al escribir un programa no lo hagamos sobre alguno anterior que esté colocado en una determinada dirección. Sin embargo, con tantos Ks de memoria RAM como dispone nuestro ordenador, no va a resultarnos difícil encontrar un espacio libre.

El Programa I es un listado en Basic de la utilidad y el Programa II es otro listado de la misma, pero esta vez en lenguaje ensamblador. Elija el que mejor le parezca.

Cuando los ejecute verá como se le añaden algunos nuevos comandos al Basic de su **Amstrad.** En la Tabla I le proporcionamos una lista de los mismos.

Analicemoslos con detenimiento. El primero es:

IPRINT.PAGE

Nos vas a imprimir el valor de una de las variables del sistema que hemos llamado PAGE. Está en la dirección &AE64 si posee el Basic 1.1, en la &AE81 si tiene el Basic 1.0.

El segundo que nos encontramos es:

ISET.PAGE, entero

y, como probablemente ya habrá adivinado, establece el valor que va a tener la variable PAGE.

Cuando introduzca en memoria un programa o lo cargue desde una cinta o un disco, el código generado por las instrucciones Basic se almacenará a partir de la posición siguiente a la indicada por PAGE. Si escribimos el comando SAVE, el ordenador nos va a salvar el programa que se encuentre en la memoria colocado donde nos indique PAGE. Y lo mismo ocurre si tecleamos RUN. Lo que se nos va a ejecutar es el señalado por la variable PAGE.

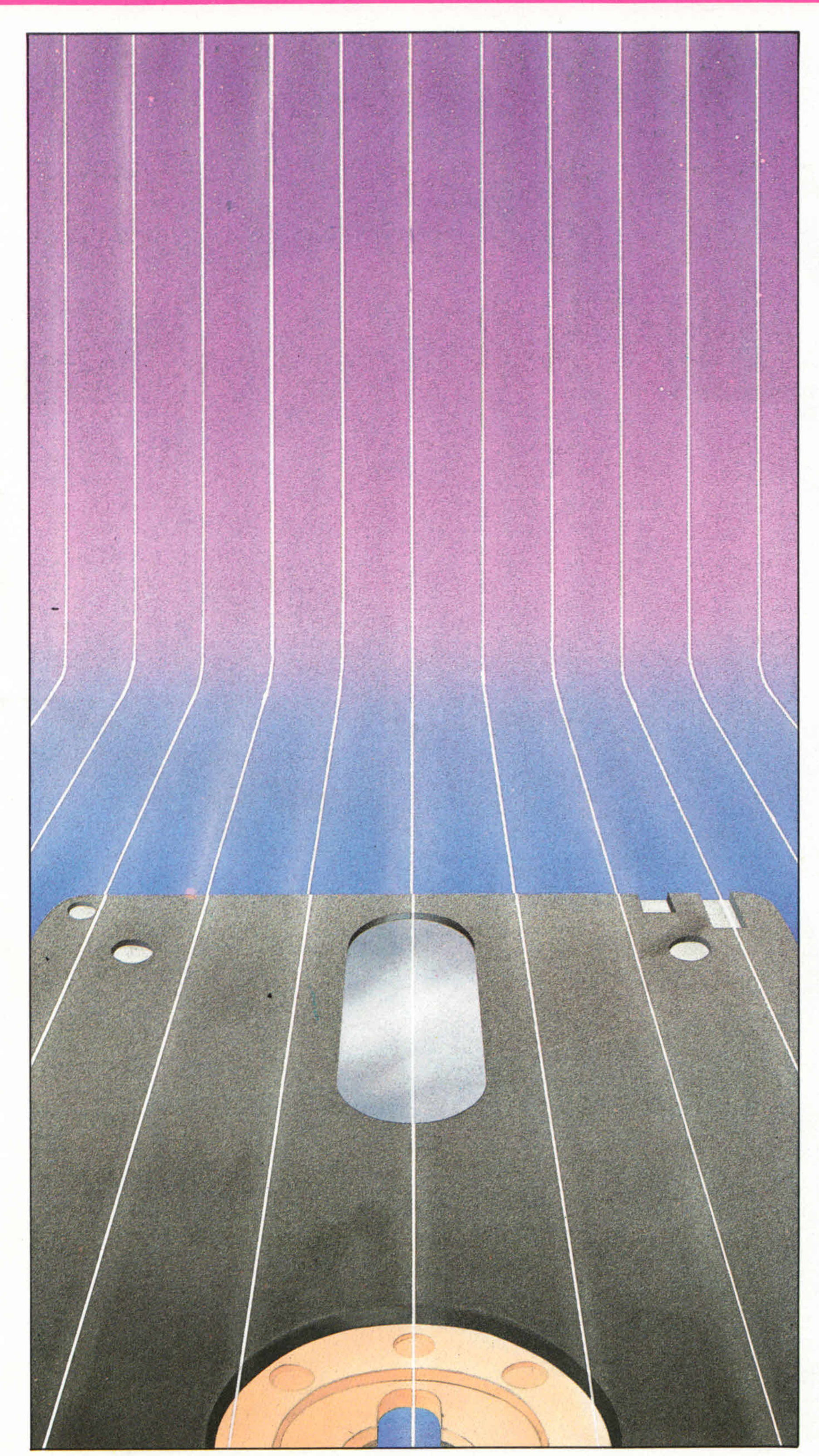
Si cambiamos su valor, podemos meter en memoria otro programa sin alterar el que ya teníamos en la misma siempre que, naturalmente, no lo escribamos encima.

No es cuestión más que de cambiar una de las variables del sistema. Si queremos volver a nuestro programa primitivo y devolver a page su valor original, no podremos hacerlo mágicamente. Es necesario que se lo indiquemos correctamente al ordenador.

Además de conocer dónde comienza un programa, en Basic también puede sernos necesario saber dónde termina. Esta dirección almacena, igualmente, en otra de las variables del sistema a la que llamamos TOP.

El Basic necesita conocer su valor, ya que a partir de esta posición es donde se almacenan las variables cuando el programa se está ejecutando.

También van a ser muy importantes otro tipo de apuntadores. Son los que le van a decir al interpretador Basic el punto donde termina la zona de memoria destinada a contener las distin-



tas variables que, como anteriormente dijimos, van a estar situadas al final del programa a partir de la dirección señalada por TOP.

Al resultar todos estos apuntadores, restablecemos al mismo tiempo el programa original, que podemos ejecutar otra vez.

Las variables, sin embargo, se perderán, y le aconsejamos que teclee CLEAR siempre que cambie el valor de PAGE.

La utilidad realiza todo esto automáticamente. Así que si alteramos PAGE, la rutina examina si hay almacenado algún programa en la nueva dirección. Si no hay ninguno, o no podemos elegir el programa existente debido a que lo hayamos «manchado», se ejecuta inmediatamente el comando:

INEW.PROGRAM

Esta orden suprimirá el programa que haya en la dirección señalada por PAGE sin destruir ninguno de los otros que comparten en ese momento la memoria. La instrucción Basic NEW limpia todo lo que haya en la misma, así que utilícela con cuidado.

Un programa puede necesitar conocer en un determinado momento que PAGE, TOP y LOMEM tiene. Veamos qué es esta última palabra: LOMEM es la dirección más baja de la zona de memoria que está libre. ¿Correcto?

Pero, ¿cómo vamos a saber el valor de cada una de estas tres variables? La utilidad PAGE nos va a proporcionar tres nuevos comandos para que lo podamos hacer con toda facilidad.

IGET.PAGE
IGET.TOP
y
IGET.LOMEM

toman el valor actual de cada una de estas variables del sistema y lo coloca en una, definida como entera, del lenguaje Basic.

Si hacemos:

a%=0

y a continuación:

IGET.PAGE,¶a%

la variable entera contendrá ahora el valor que tiene en este momento PAGE.

Podemos hacer los mismo para TOP y LOMEM. ¿Se atreve?

''PAGE'' es una utilidad extremadamente usada y de una gran ayuda a la hora de programar. Nos permite conseguir todas las ventajas de la generosa cantidad de memoria RAM que posee nuestro **Amstrad**.

Anímese. Coja una de nuestras revistas y seleccione varios programas pequeños. Cargue cada uno de ellos en una posición seleccionada por el comando:

ISET.PAGE

siguiendo las indicaciones que le hemos dado y se encontrará con que la memoria de su ordenador está compartida por todos ellos y utilizada con un rendimiento bastante mayor. ¡Suerte!

```
IPRINT.PAGE
ISET.PAGE,integer
IGET.PAGE,¶variable%
IGET TOP,¶variable%
IGET.LOMEM,¶variable%
INEW.PROGRAM
```

Tabla 1. Comandos nuevos

```
10          ORG  #A000
20 ;
30 ;***** INICIALIZAR RSX *****
40 ;
50          LD   HL,flags
60          BIT  1,(HL)
70          RET  NZ
80          SET  1,(HL)
90          LD   BC,tablasal
100         LD   HL,workspace
110         CALL #BCD1
120         CALL #B900
130         PUSH AF
140         LD   A,(#C002)
150         AND  A
160         JR   Z,inicializar
170         LD   HL,#AE64
180         LD   (PAGIN1),HL
190         LD   (PAGIN2),HL
200         LD   (PAGIN3),HL
210         LD   (PAGIN4),HL
220         LD   HL,#AE17
230         LD   (DATO1),HL
240         LD   (DATO2),HL
250         LD   HL,#AE5E
260         LD   (HIMEM),HL
270         LD   HL,#AE66
280         LD   (SUP1),HL
290         LD   (SUP3),HL
300         LD   HL,#AE68
310         LD   (SUP2),HL
320         LD   HL,#AE6A
330         LD   (ARRAY),HL
340         LD   HL,#AE6C
350         LD   (MEMAL1),HL
360         LD   (MEMAL2),HL
370         LD   HL,#AE1D
380         LD   (EJELIN),HL
390 inicializar:
400         POP  AF
410         CALL #B90C
420         CALL cadena
430         DEFB "U","t","i"
440         DEFB "l","i","d"
450         DEFB "a","d"," "
460         DEFB "P","A","G"
470         DEFB "E"," ","a"
480         DEFB "c","t","i"
490         DEFB "v","a","d"
500         DEFB "a",".",7
510         DEFB 13,10,0
520         RET
530 ;
540 ;* DAR NUEVO VALOR A PAGE *
550 ;
560 pagina:
570         DEC  A
580         JP   NZ,errorpar
590         CALL chequeo
600         LD   E,(IX+0)
610         LD   D,(IX+1)
620         LD   HL,(#AE7B)
630 HIMEM:  EQU  $-2
640         DEC  H
650         PUSH HL
660         AND  A
670         SBC  HL,DE
680         JP   C,nositio
690         LD   HL,#016C
700         SBC  HL,DE
710         EX   DE,HL
720         POP  DE
730         JP   NC,nopuedo
740         LD   (#AE81),HL
750 PAGIN1: EQU  $-2
760         LD   (#AE30),HL
770 DATO1:  EQU  $-2
780         LD   A,(HL)
790         AND  A
800         JR   NZ,nueva
810         INC  HL
820 linueva:
830         LD   C,(HL)
840         INC  HL
850         LD   B,(HL)
860         LD   A,B
870         AND  A
880         JR   NZ,nueva
890         OR   C
900         JR   Z,varpointer
910         DEC  HL
920         ADD  HL,BC
930         PUSH HL
940         AND  A
950         SBC  HL,DE
960         POP  HL
970         JR   C,linueva
980 ;
990 ;********** NEW **********
1000 ;
1010 nueva:
1020        LD   HL,(#AE81)
1030 PAGIN2: EQU $-2
1040        LD   (#AE30),HL
1050 DATO2: EQU  $-2
1060        LD   (HL),#00
1070        INC  HL
1080        LD   (HL),#00
1090        INC  HL
1100        LD   (HL),#00
1110 varpointer:
1120        INC  HL
1130        LD   (#AE83),HL
1140 SUP1:  EQU  $-2
1150        LD   (#AE85),HL
1160 SUP2:  EQU  $-2
1170        LD   (#AE87),HL
1180 ARRAY: EQU  $-2
1190        LD   (#AE89),HL
1200 MEMAL1: EQU $-2
1210        CALL cadena
1220        DEFB "o","k",13
1230        DEFB 10,7,0
1240        RET
1250 ;
1260 ;* PONER PAGE EN VARIABLE *
1270 ;
1280 cogerpag:
1290        DEC  A
1300        JP   NZ,errorpar
1310        LD   DE,(#AE81)
1320 PAGIN4: EQU $-2
1330 cp:
1340        LD   L,(IX+0)
1350        LD   H,(IX+1)
1360        LD   (HL),E
1370        INC  HL
1380        LD   (HL),D
1390        RET
1400 ;
1410 ;* PONER TOP EN VARIABLE *
1420 ;
1430 superior:
1440        DEC  A
1450        JP   NZ,errorpar
1460        LD   DE,(#AE83)
1470 SUP3:  EQU  $-2
1480        JR   cp
1490 ;
1500 ;* PONER LOMEM EN VARIABLE *
1510 lomem:
1520        DEC  A
1530        JP   NZ,errorpar
1540        LD   DE,(#AE89)
1550 MEMAL2: EQU $-2
1560        JR   cp
1570 ;
1580 ;***** ESCRIBIR CADENA *****
1590 ;
1600 cadena:
1610        POP  HL
1620 sp1:
1630        LD   A,(HL)
1640        CALL #BB5A
1650        INC  HL
1660        OR   A
1670        JR   NZ,sp1
1680        JP   (HL)
1690 ;
1700 ;***** ESCRIBIR PAGE *****
1710 ;
1720 escribir:
1730        LD   A,#26
1740        CALL #BB5A
1750        LD   HL,(#AE81)
1760 PAGIN3: EQU $-2
1770 ;
1780 ;** ESCRIBIR PALABRA HEX **
1790 ;
1800 hexpal: LD  A,H
1810        CALL hex
1820        LD   A,L
1830 ;
1840 ;*** ESCRIBIR BYTE HEX ***
1850 ;
1860 hex:   PUSH AF
1870        RRCA
1880        RRCA
1890        RRCA
1900        RRCA
1910        CALL hex1
1920        POP  AF
1930 hex1:  AND  #0F
1940        ADD  A,#90
1950        DAA
1960        ADC  A,#40
1970        DAA
1980        JP   #BB5A
1990 ;
2000 ;******** ERRORES ********
2010 ;
2020 nopuedo:
2030        CALL cadena
2040        DEFB "N","o"," "
2050        DEFB "p","u","e"
2060        DEFB "d","o"," "
2070        DEFB "h","a","c"
2080        DEFB "e","r","l"
2090        DEFB "o","!",13
2100        DEFB 10,7,0
2110        RET
2120 ;
2130 nositio:
2140        POP  HL
2150        CALL cadena
2160        DEFB "N","o"," "
2170        DEFB "h","a","y"
2180        DEFB " ","s","i"
```

```
2190            DEFB "t","i","o"
2200            DEFB ".",13,10
2210            DEFB 7,0
2220            RET
2230 ;
2240 error:     CALL cadena
2250            DEFB "E","r","r"
2260            DEFB "o","r"," "
2270            DEFB "R","S","X"
2280            DEFB 13,10,7
2290            DEFB 0
2300            RET
2310 ;
2320 chequeo:
2330            LD   HL,(#AE36)
2340 EJELIN:    EQU  $-2
2350            LD   A,H
2360            OR   L
2370            RET  Z
2380            POP  HL
2390            CALL cadena
2400            DEFB "P","r","o"
2410            DEFB "g","r","a"
2420            DEFB "m","a"," "
2430            DEFB "e","n"," "
2440            DEFB "e","j","e"
2450            DEFB "c","u","c"
2460            DEFB "i","o","n"
2470            DEFB "!",13,10
2480            DEFB 7,0
2490            RET
2500 ;
2510 errorpar:
2520            CALL cadena
2530            DEFB "O","l","v"
2540            DEFB "i","d","o"
2550            DEFB " ","d","e"
2560            DEFB " ","P","A"
2570            DEFB "G","E","!"
2580            DEFB 13,10,7
2590            DEFB 0
2600            RET
2610 ;
2620 ;**** TABLA DE SALTOS ****
2630 ;
2640 tablasal:
2650            DEFW nomtabla
2660            JP   escribir
2670            JP   nueva
2680            JP   pagina
2690            JP   cogerpag
2700            JP   superior
2710            JP   lomem
2720 ;
2730 ;**** TABLA DE NOMBRES ****
2740 ;
2750 nomtabla:
2760            DEFB "P","R","I"
2770            DEFB "N","T","."
2780            DEFB "P","A","G"
2790            DEFB #C5
2800            DEFB "N","E","W"
2810            DEFB ".","P","R"
2820            DEFB "O","G","R"
2830            DEFB "A",#CD
2840            DEFB "S","E","T"
2850            DEFB ".","P","A"
2860            DEFB "G",#C5
2870            DEFB "G","E","T"
2880            DEFB ".","P","A"
2890            DEFB "G",#C5
2900            DEFB "G","E","T"
2910            DEFB ".","T","O"
2920            DEFB #D0
2930            DEFB "G","E","T"
2940            DEFB ".","L","O"
2950            DEFB "M","E",#CD
2960            DEFB #00
2970 ;
2980 workspace:
2990            DEFW #00
3000            DEFW #00
3010 ;
3020 flags:
3030            DEFB #0
```

```
10 REM UTILIDAD "PAGINA"
20 REM DE R.A.WADDILOVE
30 REM (c) MICROHOBBY AMSTRAD
40 REM CALL &A000 para ???????
50 MEMORY &9FFF:direccion=&A000
60 FOR i=1 TO 44
70 suma=0:READ codigo$,chequeo$
80 FOR j=1 TO 21 STEP 2
90 byte=VAL("&"+MID$(codigo$,j,2))
100 POKE direccion,byte
110 suma=suma+byte:direccion=direcc
ion+1
120 NEXT j
130 IF suma<>VAL("&"+chequeo$) THEN
 PRINT "ERROR DE DATOS EN LA LINEA
";140+i*10
140 NEXT I
150 DATA 21E7A1CB4EC0CBCE0199A1,656
160 DATA 21E3A1CDD1BCCD00B9F53A,6B4
170 DATA 02C0A728422164AE229CA0,464
180 DATA 22BAA02214A122E4A02117,431
190 DATA AE229FA022BDA0215EAE22,4DD
200 DATA 87A02166AE22C9A022F6A0,59F
210 DATA 2168AE22CCA0216AAE22CF,4EF
220 DATA A0216CAE22D2A02200A121,453
230 DATA 1DAE2267A1F1CD0CB9CD04,549
240 DATA A15061676520557469 6C69,445
250 DATA 7479206F6B2E070D0A00C9,2FC
260 DATA 3DC285A1CD66A1DD5E00DD,611
270 DATA 56012A7BAE25E5A7ED52DA,574
280 DATA 45A1216C01ED52EBD1D22F,570
290 DATA A12281AE2230AE7EA72014,44B
300 DATA 234E234678A7200CB12817,315
310 DATA 2B09E5A7ED52E138ED2A81,5B0
320 DATA AE2230AE3600233600 2336,296
330 DATA 00232283AE2285AE2287AE,422
340 DATA 2289AECD04A16F6B0D0A07,3C3
350 DATA 00C93DC285A1ED5B81AEDD,642
360 DATA 6E00DD66017323 72C93DC2,482
370 DATA 85A1ED5B83AE18EC3DC285,627
380 DATA A1ED5B89AE18E2E17ECD5A,6A0
390 DATA BB23B720F8E93E26CD5ABB,5DC
400 DATA 2A81AE7CCD1BA17DF50F0F,4EE
410 DATA 0F0FCD24A1F1E60FC69027,513
420 DATA CE4027C35ABBCD04A14361,523
430 DATA 6E277420646F2074686174,3CD
440 DATA 210D0A0700C9E1CD04A14E,3A9
450 DATA 6F20726F6F6D0D0A0700C9,333
460 DATA CD04A15253582065 72726F,447
470 DATA 720D0A0700C92A36AE7CB5,398
480 DATA C8E1CD04A150726F677261,586
490 DATA 6D2072756E6E696E67210D,3BC
500 DATA 0A0700C9CD04A1466F7267,3DA
510 DATA 6F7420504147452 10D0A07,25F
520 DATA 00C9ADA1C30EA1C3B9A0C3,668
530 DATA 79A0C3DEA0C3F0A0C3FAA0,80A
540 DATA 5052494E542E504147C54E,3A6
550 DATA 45572E50524F475241CD53,3B5
560 DATA 45542E504147C54745542E,372
570 DATA 504147C54745542E544FD0,41E
580 DATA 4745542E4C4F4D45CD0000,308
```

Prueba comparativa de Basic's

El Basic es, sin duda alguna, el idioma más hablado por los ordenadores personales. Esto debería significar que un programa realizado con cualquiera de ellos, valdría sin más para los demás. Pero como todos sabemos esto dista de ser una realidad. Hemos probado algunos de los ordenadores más vendidos del mercado, para comparar sus «Basic's» y ver sus peculiaridades.



os ordenadores elegidos son, además del **Amstrad** CPC, el Spectrum, el QL, un MSX, el Commodore, y como punto de comparación hemos elegido el Basic Microsoft que usan multitud de ordenadores, probando su velocidad en dos de los más famosos: el IBM PC y el APPLE MacIntosh.



Resulta evidente que estos dos ordenadores están a un precio superior a los demás, (por lo menos hasta que **Amstrad** compre las fábricas de IBM y APPLE, cosa que nunca se sabe...) pero es conveniente incluirlos como punto de referencia, dado que el Basic que utilizan es el que más se acerca a un estándar.

Otra advertencia que hay que hacer, básicamente para las pruebas de velocidad, es que tanto el IBM como el MacIntosh y el QL son distintos a los demás a nivel de Hardware, y entre otras cosas el microprocesador es distinto del Z-80 que llevan los demás, lo que ha influido en muchos aspectos, sobre todo en el aspecto de la velocidad, llevándonos alguna vez a resultados curiosos, por no decir sorprendentes...

Para la realización de este estudio, se han incluido unos bancos de pruebas pensados para calcular el tiempo que tarda cada Basic en realizar alguna de las cosas más comunes dentro de un programa. Pero hay que advertir que la validez de las pruebas de velocidad es relativa. Tienen el valor que se les quiera dar, ni más ni menos, pero se han incluido para tener una referencia más, entre otras que también se proporcionan. Pero no busquen una clasificación final. No la hay. Simplemente encontrarán las características principales de cada uno de ellos, en qué aspectos destacan, y en cuáles fallan. Si uno es mejor o peor que otro es cuestión de las necesidades de cada usuario, y en consecuencia de lo que éste le vaya a pedir.

Bancos de prueba

Se han realizado a cada ordenador cinco pruebas distintas, cuyos listados aparecen en el cuadro adjunto. El primer banco de pruebas mide el tiempo que tarda cada ordenador en realizar una operación matemática que incluye funciones trigonométricas. Se han elegido este tipo de funciones con una razonable dosis de mala uva. La realización de operaciones trigonométricas exige al ordenador la creación y manejo (internamente) de unas tablas, por lo que el tiempo que se tarda es mayor que en otro tipo de operaciones.

La segunda prueba que se les ha puesto es para medir el tiempo que tarda en realizar operaciones de lectura y escritura en disco. Esta prueba sólo se ha hecho en el **Amstrad,** el IBM PC y el MacIntosh, que son los únicos que llevan dicha unidad incorporada. En los demás, dado que realizarla con cassette era claramente injusto, y que en algún caso aunque se puedan conectar unidades de distintas marcas y sistemas, éstas no van incluidas en el **«equipo básico»** del mismo, hemos decidido liberarles de esta prueba. La prueba en sí, consiste en crear un fichero secuencial, grabar doscientos datos, cerrar el fichero, volverlo a abrir, leer los doscientos datos, presentarlos en pantalla y cerrar el fichero.

En la tercera prueba se han gastado las dosis que quedaban de mala uva. Se trata de ordenar alfabéticamente por el conocido y lento método de la burbuja, quince matrices divididas en tres grupos, en cada uno de los cuales las matrices eran muy parecidas. El trabajo con matrices alfanuméricas es otra de las facetas que los ordenadores suelen manejar con cierta lentitud.

La cuarta prueba tiene dos objetivos. El primero es medir el tiempo que tarda el ordenador en realizar un bucle, y el segundo es verificar la exactitud de las operaciones numéricas. El programa consiste en sumar mil veces 0,001. Esto, naturalmente da como resultado 1. Ahora bien, algunos ordenadores utilizan un método de cálculo interno muy peculiar que hace que el resultado de esta operación se aproxime mucho a uno, sin serlo. Se ha evitado el habitual redondeo, para comprobar el resultado exacto.

La última prueba mide el tiempo que tarda el ordenador en llenar la pantalla con texto y después borrarla. A pesar de las diferentes maneras que tienen los ordenadores de gestionar la pantalla, aquí los resultados no han sido ni divertidos ni sorprendentes. Más bien han sido normales. Pero pasemos a comentar los resultados de cada prueba. Que cada ordenador se levante al oír su nombre, para escuchar los resultados de sus pruebas. Los que suspendan algún parcial, podrán volverse a examinar en septiembre. (Vaya por Dios, ya me he liado. En qué estaría yo pensando...).

1) Velocidad de cálculo

Los resultados en esta prueba han sido por lo menos, curiosos. El QL de Sinclair demuestra ser el más rápido con diferencia, mientras el MacIntosh logra un pésimo penúltimo puesto, aventajando sólo al Spectrum. (Ya podrá...). El **Amstrad** logra un honorable tercer lugar, a dos décimas de segundo de todo un IBM. Commodore y MSX hacen unos tiempos medios.

2) Lectura y escritura de ficheros

Aquí sólo se le ha hecho la prueba a los tres ordenadores que llevan unidad de disco integrada, y los resultados son de lo más norma-

les. El **Amstrad** queda descolgado de los IBM y MacIntosh, cuya tecnología es mucho más avanzada, y su sistema operativo más moderno.

3) Ordenación por el método de la burbuja

Esta prueba nos depara otra sorpresa. Los tiempos en todos los ordenadores son similares excepto en uno. Y resulta curioso porque habiendo dos ordenadores de la misma marca en estas pruebas, no es el más barato el peor, no, sino el otro. El QL, que hace un tiempo sensiblemente superior a los demás, catorce segundos frente al más rápido. El **Amstrad,** sólo es superado por **«los caros»,** IBM y MacIntosh.

4) Cálculo numérico

Como hemos dicho, esta prueba pretende determinar tanto la velocidad como la precisión a la hora de hacer determinados cálculos numéricos. En cuanto a la velocidad, el que más se sale de la media es el Commodore, y un poco menos el Spectrum. Los demás hacen los tiempos que se puede esperar de ellos, quedando el **Amstrad** muy bien clasificado, en la segunda posición, sólo superado por el MacIntosh.

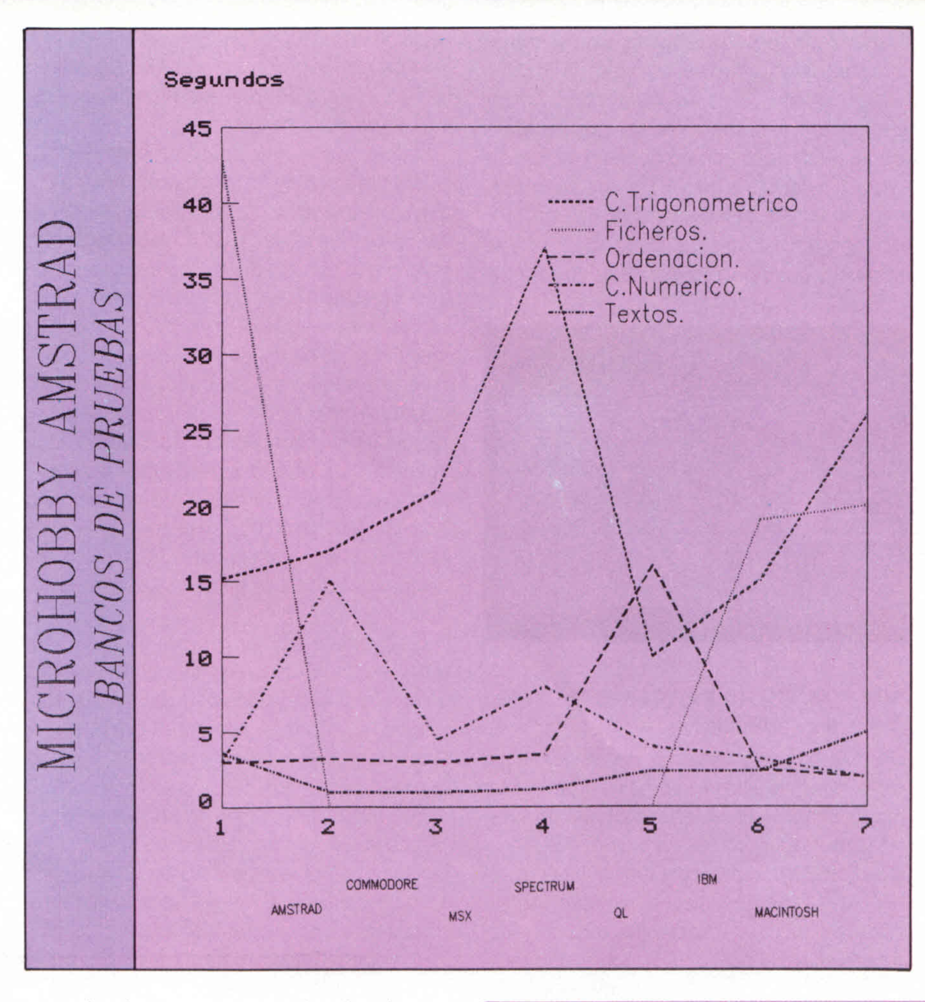
Y en precisión, tres ordenadores no dieron el resultado deseado, que era 0. Estos son **Amstrad,** Commodore y Spectrum. Aquí hay que insistir que esto se debe al sistema interno de cálculo de cada uno, y el redondeo o truncación que hacen a partir de un número de decimales. No tiene tanta importancia como parece.

5) Rapidez en presentación de textos

Esta prueba presenta dos grupos de resultados. El primero, los rápidos, compuesto por Commodore, MSX, Spectrum, QL y IBM. El segundo, los lentillos, compuesto por MacIntosh y **Amstrad.** No obstante, se está hablando de una diferencia máxima de cuatro segundos entre el más rápido y el más lento, lo que resulta poco apreciable.

6) Media de resultados

Si se observa la media realizada por cada ordenador en las cuatro pruebas que han realizado todos, el más veloz es el IBM. Pero sorprendentemente, el **Amstrad** resulta ser más veloz que los demás, en promedio. Naturalmente esto no quiere decir que sea el mejor, ni que los demás sean muy malos. Simplemente, el Basic Locomotiv resulta ser un Basic francamente rápido. En cuanto a la diferencia de tiempo entre el IBM y el MacIntosh, que usan el mismo Basic, se debe seguramente más a razones de Hardware que de Software. El QL se sitúa en una posición media, que no dice nada a favor ni en contra de él.



7)]Conclusiones?

Cada cual a la vista de los resultados, puede sacar las suyas. Incluso se podrían poner otro tipo de pruebas, y seguramente los resultados no serían los mismos. Esta es la razón por la cual este tipo de pruebas cuentan con bastante oposición. Pero la realidad es que tampoco nadie se pone de acuerdo para buscar un método mejor.

Repaso a las instrucciones de los distintos Basic's

Además de lo dicho hasta ahora, vamos a darle un repaso a las posibilidades que tienen los distintos Basic's, expresadas por la cantidad de instrucciones que cada uno de ellos poseen y sus posibilidades, y divididas por grupos, según su función.

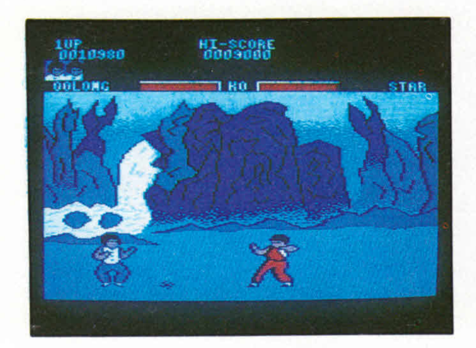
1) Comandos de manejo de ficheros

Se pretende saber las posibilidades que tiene cada ordenador para manejar ficheros, y el alcance de las mismas.

El MSBasic del IBM y MAC, fueron casi a tope, y de manera similar en ambos. Posee comandos para operar con ficheros secuenciales y aleatorios, permitiendo en los secuenciales añadir registros en ficheros ya existentes. Una opción interesante en el IBM, es que se puede tener abierto un fichero de dos maneras diferentes con distinto número con lo que se puede acceder simultáneamente de manera secuencial y al azar, o secuencialmente para entrada y salida de datos. Permite tener hasta quince ficheros abierto a la vez. No gestiona ficheros indexados, aunque por otra par-

te en ordenadores de este precio para abajo, creo que sólo lo hace el nuevo PCW 8256.

El Locomotiv Basic de **Amstrad** maneja simplemente los ficheros secuenciales sencillitos. A pesar de la unidad de disco integrada en el CPC 664 (R.I.P.) y 6128, no tiene previstos comandos para usar ficheros aleatorios, teniendo que recurrir a una rutina en código máquina, que no es lo que se puede llamar



«un modelo de perfección». En resumen, muy mejorable.

El QL, flamante demostración de tecnología de la mano de Sinclair, posee su propio sistema de almacenamiento. Son los Microdrives, pequeños cartuchos de cinta magnética. Aun siendo más rápidos y cómodos que el cassette, y teniendo en cuenta que viene con dos unidades incluidas en la máquina, cosa muy útil, no se aproxima ni a la velocidad ni a la efectividad del disco. Tampoco puede usar ficheros de acceso aleatorio. Los comandos que ofrece para la gestión de los ficheros secuenciales incluyen la posibilidad de modificar datos ya grabados. Por esta parte, bien. El resto, normal.

En el Commodore sí está prevista la instalación de unidad de disco a nivel de Basic, además de los habituales de ficheros secuenciales. Su manejo y sintaxis son comunes, no ofreciendo ninguna complicación.

El Basic resto de los ordenadores sólo tienen previsto el uso de ficheros secuenciales. Al conectar la unidad de disco, el sistema operativo del mismo debe proporcionar dichos comandos, de una u otra manera, pero en el Basic original de la máquina, ni rastro.

2) Comandos gráficos

El MSBasic dibuja en el IBM PC con dos posibles resoluciones. 320×200 puntos con 4 colores, y 640×200 puntos con 2 colores. Dispone de once comandos para el dibujo de gráficos. Permite definir las coordenadas de dibujo en cifras absolutas y relativas al último punto dibujado, pero no ofrece SPRITES. El resultado final bueno, pero mejorable.

En cuanto al MacIntosh, su sistema de dibujo es muy bueno, pero diferente. Se hace recurriendo al firmware, pero no poniendo la dirección de memoria que dibuja el gráfico, sino su nombre. Así resulta más fácil de utilizar, y la calidad lograda es grande, aunque en blanco y negro dado que el monitor que lleva incorporado no es en color.

El Basic Locomotiv dispone de varias instrucciones para la realización de gráficos en 3 anchuras de pantalla. El número de colores simultáneos depende de ésta, oscilando entre ocho y dos colores dentro de una paleta de 26. La gestión de estos colores resulta un poco liosa, sobre todo al principio, pero permite multitud de cosas, como rellenar dibujos... En gráficos, diversos comandos permiten realizar una gama de dibujos muy amplia, así como definir las coordenadas absoluta y relativamente. Quizá el único **«pero»** que se le puede poner es que no cuenta con ninguna función ya definida para realizar algún gráfico en concreto, *(por ejemplo CIRCLE...)* ni definición directa de SPRITES. A pesar de esto, buena nota.

El QL puede dibujar con ocho o cuatro colores, según la resolución que se use. Admite 512×256 puntos y 256×256 puntos. Tiene un amplio repertorio de comandos gráficos, y es bastante fácil de entender y usar. Hay que darle muy buena nota.

El Commodore es famoso por sus gráficos. Se pueden programar hasta ocho SPRITES, dispone de una paleta de dieciséis colores, alta y baja resolución, y... dos **«hermosas»** instrucciones para manejar todo esto: PEEK y POKE. Puede ser cuestión de gustos, pero parece complicado de manejar sin el manual.

El **«modesto»** Spectrum, tiene una paleta de ocho colores con dos niveles de brillo. Pero dentro de su modestia, el Basic se defiende en una dignísima posición, contando con instrucciones más que suficientes para sacarle partido y hacer buenos gráficos.

Y el MSX Basic posee ocho instrucciones gráficas, sprites, y una paleta de quince colores. Resulta uno de los Basic's quizá más completos en este aspecto, y además sus comandos son fáciles de entender y manejar. Muy bien en este aspecto.

3) Comandos de manejo del microprocesador

Aquí el MSBasic del IBM PC y el MacIntosh no es lo que se dice el RAMBO de los lenguajes Basic. Sólo dispone de una función de reloj, pero no incorpora ninguna función para manejar interrupciones desde el Basic. Alguno que otro le da sopas con honda.

Uno de los que le superan es el Locomotive Basic, que dispone de cuatro instrucciones distintas para programar interrupciones, que le permiten simular una gestión multiárea. Aquí el Locomotive alcanza una nota muy alta.

El MSX Basic permite controlar interrupciones a través de cuatro instrucciones. No es tan completo como el Locomotive, pero supera al resto de los ordenadores probados, por un amplio margen.

```
10 'B.P. Calculo Trigonometrico.
15 DIM res%(11)
20 FOR n=0 TO 10  STEP 1
30 FOR j=1 TO 5
40 FOR k=1 TO 5
50 DEF FNfunc= SIN(2*PI/(n+1)*j)*CO
S(2*PI/(n+1)*k)
60 res%(n+1)=res%(n+1)+FNfunc
70 NEXT k
80 NEXT j
90 PRINT "Para N= ";n; "la Funcion
vale ";res%(n+1)
100 NEXT n
```

```
10 ' Prueba Numero 2
20 REM tiempo de operacion en fiche
ros.
30 DIM x$(100),y$(100)
40 FOR a=1 TO 100
50 READ x$(a),y$(a)
60 RESTORE 230
70 NEXT a
80 OPENOUT "Pruebas.dat"
90 FOR a=1 TO 100
100 WRITE #9,x$(a),y$(a)
110 NEXT a
120 CLOSEOUT
130 OPENIN "Pruebas.dat"
140 FOR a=1 TO 100
150 INPUT #9,x$(a),y$(a)
160 NEXT
170 CLOSEIN
180 FOR a=1 TO 100
190 PRINT x$(a),y$(a),"GRABADO":
200 NEXT
210 PRINT "FIN"
220 END
230 DATA "ABCDEFGHIJ","0123456789"
```

```
10 ' Prueba Numero 3
20 ' Ordenacion por el metodo de la
 burbuja
30 DIM a$(15)
40 FOR x=1 TO 15
50 READ a$(x)
60 NEXT x
70 FOR y=1 TO 15
80 FOR z=y+1 TO 15
90 IF a$(z)<a$(y) THEN b$=a$(y): a$
(y)=a$(z):a$(z)=b$
100 NEXT z
110 NEXT y
120 FOR x=1 TO 15
130 PRINT a$(x)
140 NEXT x
150 DATA "AAAAAAAAAAAAAAAAAAAB","AA
AAAAAAAAAAAAAAAAA1","AAAAAAAAAAAAAA
AAAAA@","AAAAAAAAAAAAAAAAAAA ","AAA
AAAAAAAAAAAAAAAA"
160 DATA "12345abcdefg!{^ADS$A","A1
a1b@Bc$Cd2D#34%e*:","12345abcdegf{!
ADSA$]","12A34a%&56$%+:1234B*","A$S
DA^{!gfedcba54321"
170 DATA "12345678909876554321","13
579086421357908642","13246579808978
675645","1234567890-=+1234567","123
4567890987654321"
```

```
10 'Exactitud Numerica
20 a=0
30 FOR x=1 TO 1000
40 a=a+0.001
50 NEXT x
60 a=a-1
70 PRINT a
```

```
10 'Prueba de texto
20 MODE 1
30 a$=STRING$(40,"*")
40 FOR x=1 TO 25
50 PRINT a$;
60 NEXT x
70 CLS
80 END:
```

El resto de los ordenadores probados no permite realizar interrupciones a través del Basic. Ni rastro de comandos de este tipo.

4) Comandos de gestión de cadenas y matrices

Aquí sí. Excepto convertir una cadena en una salchicha con queso, el MSBasic que usan MacIntosh e IBM lo hace todo. O por lo menos, todo lo que se ha inventado hasta la fecha. Sin más comentarios.

Aunque el Locomotiv Basic no le anda a la zaga. Posee las mismas instrucciones, y con un formato prácticamente igual. Ambos lenguajes son prácticamente igual de buenos en este aspecto.

El MSX se une a los dos anteriores, poseyendo un amplio repertorio de instrucciones para el manejo de cadenas y matrices. Estas son además casi iguales a las de los anteriores, pudiéndose hablar en este caso, de que son estándar.

El Commodore se queda corto, comparado con los anteriores. Dispone de las funciones justas, y desde luego sin ningún refinamiento. En resumen, espartano.

El Spectrum se aleja de lo común en el manejo de cadenas y matrices. Resulta limitado para manejar grandes cadenas, y difiere un poco en matrices, aunque en este aspecto se defiende mejor que en las cadenas. Regular sin más.

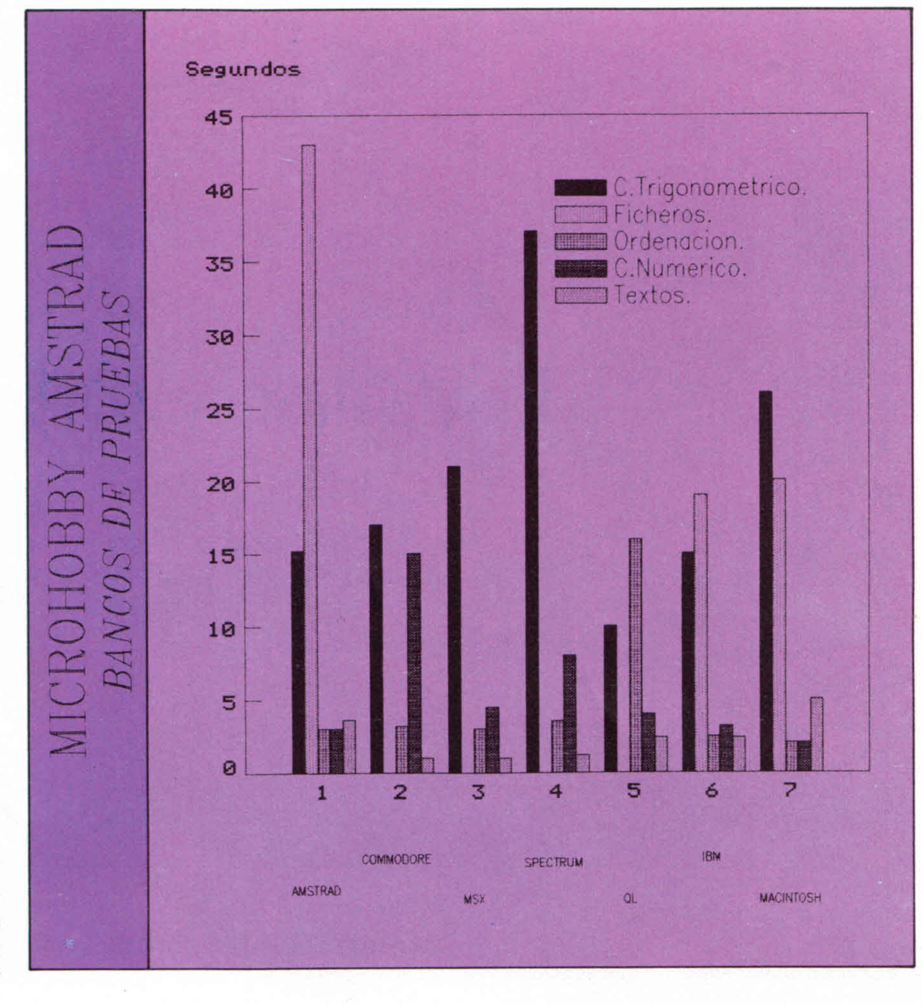
El Basic del QL también se sale de lo habitual, al igual que su hermano pequeño. Proporciona, no obstante, unas posibilidades similares a las de MSBasic. Aquí se ha esmerado Mr. Sinclair.

5) Comandos de sonido

Siete octavas y 84 notas combinables con dos instrucciones son las posibilidades sonoras del MSBasic. Así, como suena. La sentencia SOUND, que permite seleccionar la frecuencia y duración de un sonido, y la sentencia PLAY, muy buena, que permite realizar sonido a través de una cadena de caracteres, y de utilización muy sencilla. Además, resulta muy fácil de utilizar. A nivel de Hardware el IBM no proporciona una calidad tan alta como el MacIntosh, aunque éste dispone además de lo dicho, comandos en código máquina que aumentan claramente sus posibilidades.

El Locomotiv Basic de **Amstrad,** proporciona muchas posibilidades sonoras. El ordenador posee tres canales de salida de sonido, y los tres comandos disponibles permiten definir envolventes de tono, de volumen y notas. Aquí el repertorio de combinaciones posibles, suena bien.

El sonido es la otra faceta famosa del Commodore. Unas posibilidades sonoras francamente buenas, pero a que nadie adivina cómo se usan. Sorpresa: con PEEK y POKE. En este ordenador parece que todo se puede hacer con dos palabras.



La capacidad sonora de los MSX es muy amplia, poseyendo tres canales de sonido que son manejados a través de dos instrucciones y un repertorio de macrocomandos para especificar la nota, duración, pausa, ritmo, intensidad, timbre, y para asignar instrucciones al contenido de variables. Sin duda es un amplio repertorio, no es muy fácil de manejar, pero con grandes posibilidades.

Parco, como de costumbre, el Spectrum sólo tiene un comando para producir sonido, especificando duración y tono. Resulta simple de utilizar, aunque los resultados tampoco son para quedarse sordo de alegría.

Y el QL se parece mucho a su hermano pequeño. Una sola instrucción, y varios parámetros para definir duración, volumen, intervalo, etc. El balance es idéntico al del Spectrum.

6) Comandos de gestión de la pantalla. Editor

El MSBasic del IBM PC ofrece dos anchos de pantalla, que son 40 y 48 columnas. El comando window no define ventanas, sino las coordenadas de la única ventana, permitiendo hacer un efecto ZOOM a lo Valerio Lazarov. El editor es de pantalla, y muy cómodo de utilizar.

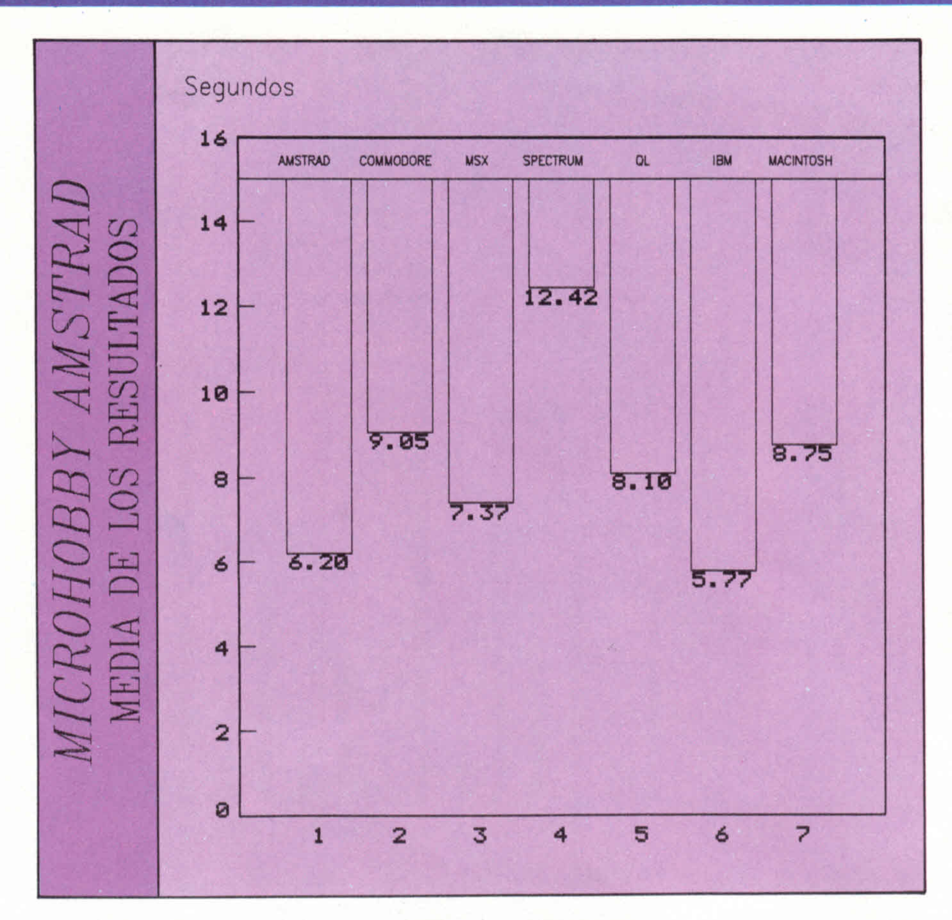
El MacIntosh posee un editor distinto, que no necesita números de línea, y opera a base de etiquetas. Además de curioso, cosa común a todas las funciones de este ordenador, muy práctico y moderno.

El Basic Locomotiv permite definir hasta ocho ventanas de texto y una de gráficos. Su editor dispone de varias funciones que facilitan realizar un texto, colocarlo adecuadamente en la pantalla, y resulta muy cómodo y fácil de utilizar.

El MSX tiene un repertorio de instrucciones, en su mayoría iguales a las de los anteriores. Su editor es normal, con bastantes funciones de edición y seguimiento de programa. Es fácil de utilizar, y proporciona un rendimiento muy bueno.

En el mundillo de los ordenadores, el editor que posee el Commodore es famoso, y no precisamente por su lado positivo. Es incómodo y complicado. Las teclas poseen varias funciones, entre ellas el cambio de color, pero esto no ayuda, sino que más bien complica aún más su manejo.

El editor del Spectrum se mantiene en la media. No es extraordinario, pero tampoco es malo. Es fácil acostumbrarse a él, una vez que se aprende el significado de los cursores. La gestión de pantalla es mejor, con instruccio-



nes fáciles de utilizar, y resultados más que aceptables.

En cuanto al QL, supongo que es cuestión de gustos, pero a mí no me gusta. No es complicado de manejar y divide la pantalla en dos ventanas, aunque esto se puede modificar, mediante la habitual instrucción WINDOW. Posee dos ajustes, según se esté utilizando un monitor o una televisión.

7) Comandos de salida a la impresora

En este aspecto, el MSBasic es sencillo como el mecanismo de un chupete, tanto en uno como en otro ordenador. La comunicación con la impresora se hace a través de instrucciones normales a las que se antepone la letra L.

El MSX, se une al anterior, usando idéntica letra para distinguir el envío de la información a la pantalla o a la impresora.

El Spectrum envía la información de igual forma que los anteriores.

El Locomotiv dirige la comunicación hacia la impresora a través del canal 8. Por esto, las instrucciones que envían la información a la impresora van seguidas de la expresión del número de canal (#8), siendo el resto idéntico a lo descrito anteriormente.

El Commodore, se une al anterior para enviar la información a la impresora a través de un canal de comunicación.

Y el QL, se une a este segundo grupo. Es necesario abrir antes un canal de comunicaciones hacia la impresora, y luego se manda la información. No es el más cómodo, pero tampoco es difícil. Un poco más largo, quizá.

8) Libro de instrucciones

La documentación ofrecida con los IBM ha hecho escuela. Perfectamente estructurada, y exhaustiva. Conociendo el Basic de otro ordenador, adaptarse a éste es fácil y muy rápido.

El manual de Basic del LOCOMOTIV es bueno sin más. Se podrían ampliar las explicaciones en ciertos apartados, pero en otros la información ofrecida es más que suficiente. Quizá se le pueda poner algún reparo a la estructuración del mismo, pero esto es cuestión de gustos.

El manual de Basic de Commodore, es pequeño. O grande, nunca se sabe, dado que con PEEK y POKE se puede hacer casi todo. Resulta escueto y un poco espartano, y su estructura es discutible. Se puede mejorar, y mucho.

Los MSX ofrecen un buen manual de instrucciones. Muy bien explicado, y con una estructura a la que no se puede poner reparos. Buena nota.

El manual del Spectrum tiene sus pros y sus contras. La información que ofrece no es exhaustiva, pero su estructura, sin embargo, es muy buena, sobre todo para aprender a programar en Basic. Con el manual en la mano quizá no se le pueda sacar todo el rendimiento al ordenador, pero desde luego se puede partir de cero, y llegar a un nivel más que aceptable.

La parte del manual del QL que corresponde al Basic, no está orientada de la misma manera que la de su hermano pequeño. Da un pequeño repaso a las posibilidades del ordenador, y una escueta información de las instrucciones del Basic. La búsqueda de una instrucción concreta no resulta fácil, debido a lo peculiar que es su distribución.

Resumen final

Si bien no hay un estándar de Basic, el más común es el Microsoft Basic, cuya versión más famosa es la usada por el IBM PC. El Apple MacIntosh lo utiliza, pero aprovecha las particularidades de su Hardware y sistema operativo para sacar más rendimiento en algunas facetas, aunque esto vaya en detrimento de otras.

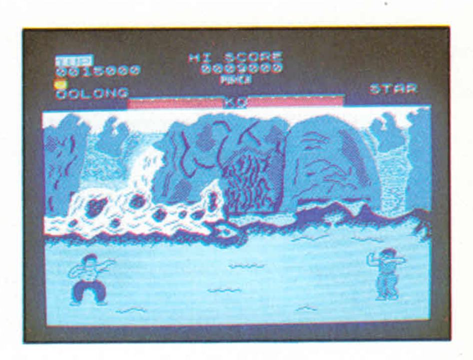
El Locomotiv Basic de **Amstrad,** se aproxima muchísimo a este modelo. Desgraciada e incomprensiblemente, el punto en el que más se aleja es en el manejo de ficheros, cuando resulta que una de sus principales características es incorporar una unidad de disco, y a un precio muy bajo. Es una pena, aunque esto quizá se compensa por la ampliación efectuada en los comandos de gráficos y sonido.

El Basic de los MSX, compatible entre ellos, es en realidad otra versión del Basic Microsoft, por lo que las similitudes también son numerosas, y prácticamente las diferencias están en los mismos puntos que los dichos para el Locomotiv, aunque entre ambos Basic's en estos apartados, no exista ninguna similitud.

El Basic del Commodore es más bien pobre. Recurre mucho a rutinas de Firmware, Peeks y Pokes, y por el resto, es más bien modesto. Además es bastante incómodo de utilizar, aunque en cuanto a calidad gráfica y sonora esté entre los mejores, si no es el mejor.

Los Basics de Sinclair son también bastante originales. Aunque no se le pueden negar sus posibilidades, se alejan de cualquier estándar, para constituir una clase aparte.

Para explicarlo todo de estos ocho Basic's se necesitarían varias revistas como ésta. Pero esperamos que esta breve revisión de ellos, les haga darse una idea de las posibilidades de todos. Y si su vecina, o su cuñada dice que el Basic de su ordenador es mejor, con el contenido de este artículo podrá demostrarle que no es así.



XIGRAMAS

Como ya sabes, tu Amstrad se puede utilizar para un montón de cosas. El nuestro nos despierta por la mañana, enseña a los niños, es una tableta para dibujar, canta para el periquito, y nos ayuda con el trabajo. Todavía no lo hemos convencido que saque los perros a paseo, pero tenemos esperanzas. El otro día intentábamos hacer un crucigrama en un periódico. Hizimos tantos errores y tanto teníamos que borrar que el periódico quedó destrozado con el crucigrama sólo medio hecho.

Entonces pensamos: ¿por qué no usamos nuestro fiel amigo (el **Amstrad,** no los perros) para hacer crucigramas en la pantalla? Nosotros somos medio españoles, medio ingleses, entonces pensamos: ¿por qué no hacemos un crucigrama también medio español, medio inglés para ayudar a nuestros amigos a aumentar su vocabulario inglés y a sus colegas de habla inglesa aprender un poco de español?

Después de tanto pensar, descansamos un rato y entonces a trabajar, y esto es lo que nos ha salido. Cada pista puede ser o en inglés o en español, o una mezcla. Si es española la pista, es inglesa la solución y viceversa. Nos divierten más las pistas escondidas, como cuando hay que convertirse en Sherlock Holmes el detective incansable para hacerse con la solución. He aquí un ejemplo:

«Mad Star es la estrella loca de los ordenadores (anag).» Estrella es STAR, loca es MAD (el adjetivo va delante en inglés). Es anagrama. Debe ser **Amstrad.**

Nuestra tarea es darte dos o más pistas hacia la solución cuando sea posible, y divertirte un poco, que no es tarea fácil. Hay una tecla de **«chuletear»:** la pulsas y sale la letra que buscabas. (Si no sabes nada de inglés todavía, sigue para ver todos los resultados. Así empiezas sin dolor a aumentar el número de palabras conectadas con los ordenadores que reconocerás.) Un crucigrama nuevo sólo requiere teclear una (sí, una) línea DATA nueva. El resultado final lo puedes pasar a impresora si tienes una con modo gráfico. Nosotros no tenemos pero lo hemos probado en la de un amigo y sale muy bonito.

No hemos oído de otros crucigramas en ordenador ni en dos idiomas a la vez. Puede ser que estos sean los primeros. Lo hemos pasado bomba preparándolos, esperamos que lo paséis bien haciéndolos.

El binario del programa base **«xigrama»** se prepara una sola vez y entonces se le usa para cada crucigrama. Al teclear las líneas DATA en el programa cargador de código máquina, usa MODE 2. Hay un simple checksum al final de cada línea: un error se señala con **«Error en xxx»** y habrá que corregir la línea xxx. No hace falta contar espacios, uno basta. Las observaciones que siguen al apóstrofe ['] pueden reemplazarse con un par de asteriscos [**]. La línea 1008 pasa el fichero binario a disco automáticamente. Nosotros usamos cassette, y cuando sale **«Press REC and PLAY etc»** encontramos mejor poner una cinta a su principio para grabar el fichero **«xigrama».** Así es fácilmente disponible para cada crucigrama.

Instrucciones

El pequeño programa en Basic es el crucigrama de esta semana. Teclea en MODE 1 porque el listado mismo está formateado a 40 columnas, y te ayudará a captar errores. Sólo la línea 10 es distinta cada vez: después del número del crucigrama vienen letras MAYUSCULAS que representan la solución codificada. Las 4 últimas son un sistema sostisficado de chequeo que dejarán pasar menos de 1 error en 50.000. La línea 30 carga el programa maestro de código máquina **«xigrama».** Entonces si usas cassette hay que tener éste preparado. RUN y cuando veas **«Press PLAY etc...»** pulsa una tecla para cargar el binario **«xigrama».** Si ves el mensaje **«Error»** en vídeo inverso, pulsa ESC/ESC, edita la línea 10, rebobina **«xigrama»,** y RUN otra vez. En el programa se mueve con las teclas cursores, y la tecla COPY saca la letra de la solución si quieres ayuda. Al terminar, **«faltan 10»** en el marcador indicaría que faltan todavía diez casitas para rellenar o corregir.

Crucigrama 1

Horizontal

2 To key-in is real crazy etc.
19 Arte inglés. ¡Eres antiguo!
25 Una x invierte la red.
30 This is among: «And before I after the train crash».
37 Se hace alquitrán la rata rebotada.
43 He laughs mid cries.
49 Suma, papá confundido.
57 No sentado y abajo ¿entendido?
64 Echarse para mentir.
70 That's it!
73 Hormiga morena tan confusa.
79 Para más U igual a cuatro.

Vertical

3 Introducir una mezcla de 30.
5 New driver is toad twisted the program on paper.
7 Después las interrupciones.
10 Coloured or green is the size of bread.
18 El precedente se halla en el siguiente ante río revuelto.
49 En sitio determinado un niño hace pino para decir «thanks».
51 Hacer re mi fa.
57 Poner juego.
61 Define the top half.
65 Yo al norte dentro.
71 ¡Así mulos!

```
10 DATA * 1@UGLZYKI@Q@N@K@P@EEZH@R@VQSO@
FPVDM@HTKL@K@ACBM@@CEG@@FO@PDGJU@LSMH@@@
JBQYQC@@@KRCUXVL*

20 hmm=HIMEM:MEMORY 5ØØØ
3Ø LOAD"xigrama",&5ØØØ:CALL &5ØØØ
5Ø CLOSEIN:MEMORY hmm
```

CRUCIGRAMA 1

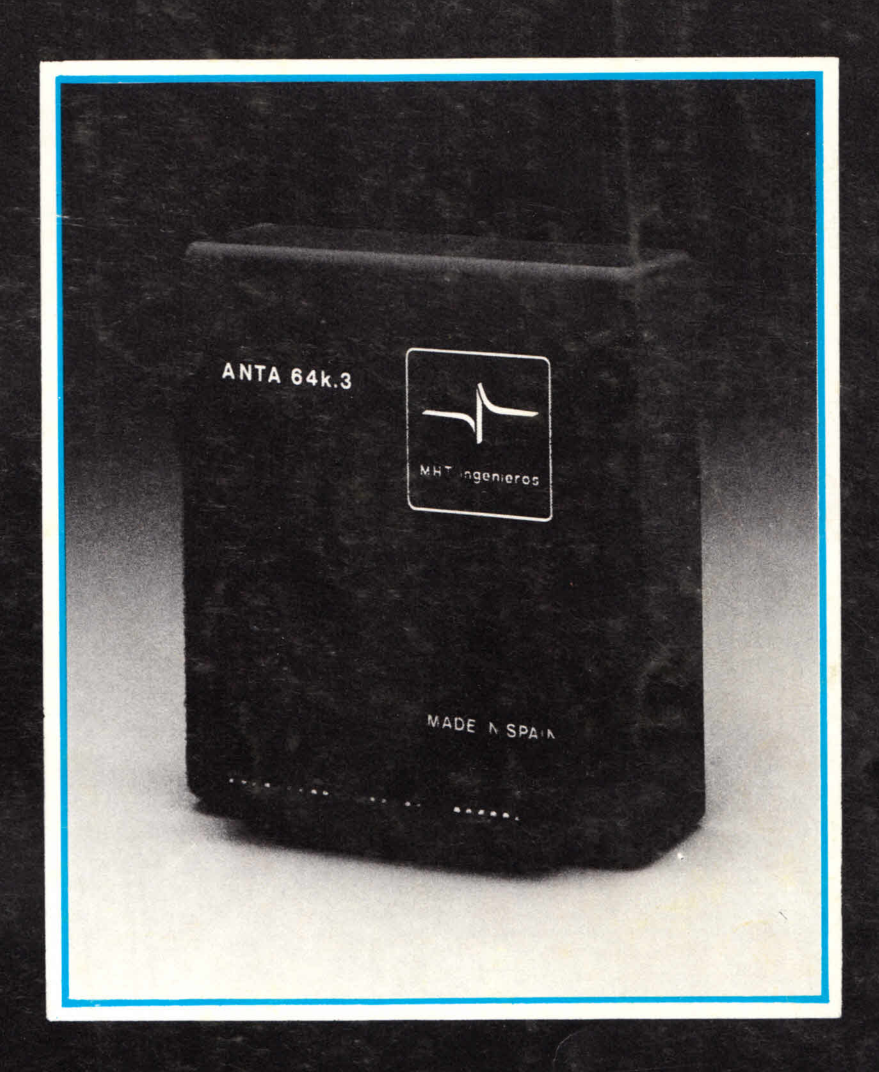
```
1ØØ DATA 211E5Ø1152ØØ15EØ1EDBØ11ØØ3ØØ1C6ØØEDBØ11ØØ15Ø1D6Ø3Ø714
11Ø DATA EDBØC34D173Ø1ØE67FC35ABBC4E3EAE8FC8B8284A3A7BEAØC6ØFB5
12Ø DATA CFD6EEE9E8FB8E8E8Ø13ØØ4CØ1ØØ91Ø18EØ1Ø1Ø8Ø27EØ1Ø21ØØ918
13Ø DATA ØØ5EØ1ØØØ9Ø24CØ1Ø191ØØ2AØØØØ168Ø8198AEA9ACA7C4BCB6Ø8Ø2
14Ø DATA 83F6E5F9E6ECF2FEC2DECØCC4D11598ØCD7Ø18EBCD7F18C3C11ØA4
15Ø DATA 8Ø214CØ1CDCØBB11ØØØØ21EØFEC3F9BBF3ED5FF53E8ØE2B718ØD6Ø
16Ø DATA EØ164F7AFEØ8C843CD39BB1518F5ØFØØ1FØ1Ø166616C74616EØ959
17Ø DATA 1F24Ø168656C7Ø73ØFØ31FØ3Ø22Ø3Ø1F25Ø22Ø3ØFF1FØ11814Ø4C7
18Ø DATA 1FØFØ2435255434947524 14D412ØØØØØFF21ØØ2DØ9473E4ØBEØ6Ø7
19Ø DATA C9AFB6FABABBCDE4BBCD3881E5C5E1CDCFBBE1CD3881E5C5E11263
2ØØ DATA CDD2BBCDDBBBE12318DE235E2356234E2346C9FEØF3811F5D5ØC74
21Ø DATA E6FØØFØF47ØFØFCD5981D113F1E6ØF47Ø7Ø78Ø21763Ø4FØ6ØØØ8BB
22Ø DATA Ø9EBØ6Ø51AB67713CD26BC1ØF7C911E8FF2186Ø1CDCØBBØ613ØAD9
23Ø DATA C5112ØØØ21EØFECDC3BB11ØØØØ212ØØ1CDF9BBC11ØEAØ1D6Ø3ØAA9
24Ø DATA 11ØØ15CD2717Ø6ØAC5E511CØFDCDF9BB21EØFF114ØØ2CDC3BBØBD8
25Ø DATA E1C11ØEBC982879394A4B7B698DFC956869E97BBA4B5D9EA961Ø65
26Ø DATA AE978185AØ91981184AAACA8C5DØ82AE889E93DFFDFFFB83D81Ø56
27Ø DATA DØDDABB1F9928A869CA6ECF7CCCFDDC6A4BC6DDFCBCB9EB7AE124C
28Ø DATA D6FØ88E1FBD1CD95CAA2A2B48B85958F999681AEA8FØF39597 1168
29Ø DATA F7EFE9829AF6ECB6FED1C4DE227E23FEFFC8CD548Ø18F62Ø5Ø1Ø9B
3ØØ DATA 5Ø5Ø2Ø2Ø6Ø2Ø2Ø2Ø7Ø1Ø3Ø4Ø7Ø7Ø1Ø2Ø1Ø7Ø1Ø3Ø5Ø7Ø1Ø7Ø4ØØ5EØ
31Ø DATA 3Ø1Ø6Ø7Ø4Ø7Ø5Ø7Ø7Ø1Ø2Ø4Ø4Ø7Ø5Ø7Ø5Ø7Ø7Ø5Ø7Ø1Ø7Ø21ØDØ76E
32Ø DATA 16CDC18Ø3EØ3CDDEBBE5CD83163EØ2CDDEBBCD4C16E1CD19BDØD6F
33Ø DATA C3168111417521512D3E81Ø6Ø9F5C5D5E5CD4181D11B21Ø153Ø9F2
34Ø DATA 193E4ØBEEB28Ø23EØØ77E3E52ØØBCD29BCØEFØ11ØFØ4CD47BCØ9B6
35Ø DATA D11B1B1B1BE1C1F1D6Ø127C81ØCDØ184FFEBØ9EB18C322ØØBFØB92
36Ø DATA 11AØØ51922Ø2BF2156C922Ø4BF19220 6BFØ691C5CD19BDCBØ8Ø8A9
37Ø DATA 3Ø162AØ4BFE5CD26BC22Ø4BF2AØØBFE5CD26BC22ØØBF18142AØ96Ø
38Ø DATA Ø6BFE5CD29BC22Ø6BF2AØ2BFE5CD29BC22Ø2BFE1D1Ø124ØØEDØB6C
39Ø DATA BØC11ØC4C997A2BØB8AØCØDB5EBCA8B3BAD8CFD7DEF4F2FB8211D8
4ØØ DATA 829491A4AABDCCCF5AB9B4E7FØ988F988Ø8FE7EBB8818A93EF1ØDØ
41Ø DATA AØA3B68E889EF68E87B6BFFBC1DBADA5A6B6C7DØD9A2CEFAFA1246
42Ø DATA CAC6CBA9EAF3FCA6819AFØFCE6F2CACA969F8BA3B59Ø9F829112 5Ø
43Ø DATA F5F493A8A3BCB2C49D99A2CAD3DCA5CCF6C4C8D8D43E89979311DA
44Ø DATA B5BAB1DFCDDED5FBF3E39897969BAØA1B8DBCAF4DFF362Ø3Ø81181
45Ø DATA Ø176Ø12ØØØ5EØ1Ø3ØØØØ7EØ292ØØECØØØ218Ø166Ø12ØØØ5EØ1Ø3F9
```

```
460 DATA Ø2ØØØØ7EØ29EØØEØØØ8Ø213EØ1229E16219EØØ22A11621ØØØØ58F
470 DATA 11ØØØØCD7F16111AØ121CØØØCD7F162162FF22A116215EØ111Ø6CE
480 DATA D8FFCD7F1621BEØØ1142Ø1Ø6141814212ØØØCD921621C2FE22Ø86B
490 DATA A116215EØ1Ø63811C8FFE5D5C5CDCØBB117EØ2213EØ1CDF9BBØB86
500 DATA C1D1E113131ØEAC9CDØD81C87711Ø15319463EØ2CD96BBCDF3ØBD8
510 DATA 16AFC396BBCD7BBBCD1BBB3ØFBFEFCC821C416E5DØFE1Ø2ØØ1ØE4B
520 DATA Ø7FE2ØØ6ØØ28ØFFEEØ28CBE6DFFAFD16FE41D8FE5BDØCDØD81ØDAØ
530 DATA C87ØCD7EBB78CD5ABB79181AD6DØD8FEØ8DØCB4F2ØØ2CBD8Ø4ØD7A
540 DATA 1F7838Ø2ED44418ØF8FE51DØ4F2EFF2CD6Ø93ØFBC6ØACB15Ø7ØB43
550 DATA 67C375BB21ØØØØD5C51AØ6Ø811Ø58Ø4FE68ØAC67293ØØ67BADØ922
560 DATA 6F7AAC67791710EEC1D113ØB78B12ØDEC911CDØ121538Ø1A1BØA32
570 DATA D649ED6728F82BE25317EBØ152ØØEDB8131A21AØ154FAF47Ø9ØA3E
580 DATA EB4FØ651233E4ØBE28121A81E61F4F13ED4486C61AFE5B38FAØA4E
590 DATA D61A771ØE611Ø18ØØ153ØØCD27177DB421ØB3ØC2C18ØCDFFBBØA65
600 DATA CD4EBB21AØ15CDC18ØE53E4ØCDØ8BCCDBABBCDDBBBCD6E81E1ØEFØ
610 DATA CDC18Ø3ECØCDØ8BCCDA83ØØ6ØØCD19BD1ØFB11BØØØCDB28Ø11ØBC7
620 DATA DØØ1CDB28Ø21Ø16ØCD4515111C3ØCD7Ø18CDØØ15CD18BB2177Ø945
630 DATA Ø111ØA81Ø1Ø2ØØEDBØ11F88ØCD6Ø1821216ØCD4515219815CDØ86F
640 DATA C18Ø114Ø3ØCD7Ø18CD81BB1EØØCDCB8ØCDØ9BB38FBØEØØCDC4ØBB9
650 DATA 16CD2C1821ØØ3ØCDC18Ø1EFFC3CB8Ø21Ø18Ø11ØØ2DØ152ØØ1AØ7FE
660 DATA EDA1E25D181328F7FE4ØE521F68Ø28Ø321F18Ø347EFE3A38ØBØBBB
670 DATA 363Ø2B7EFE2Ø2ØØ2363Ø34E118D811D88Ø216C3Ø227818CD7ØØ8CF
680 DATA 1821C18Ø227818C93EØ1CDB4BB47EBCDC18ØEB78C3B4BBAFB6ØDAA
690 DATA 23F8E5CD59BCE15E235623E5EB11D2Ø1CDCØBBE17ECDDEBB23ØEA1
700 DATA E521ØØØØ228F18118ØØ2CDF9BB21FEFF118ØFDCDC3BBE1352ØØC1Ø
710 DATA E62318C8B5DD67F1E4CA8ØF3DD7CED4FØØAE23FAD1183B3BF5ØEA8
720 DATA ED5FDD67F1C3548ØA9E7Ø6A8
1000 '
1001 hmm=HIMEM
1002 h=&5ØØØ-1
1003 MEMORY h
1006     MODE 2:PRINT"esperar 2Ø seg."
1007     direc=h:GOSUB 2ØØØ'                              cargador
1008     SAVE"xigrama",b,h+1,156Ø
1010 MEMORY hmm:END
2000 '                        rutina carga datos con checksum
2010 FOR linea=1ØØ TO 72Ø STEP 1Ø
2020   READ postizo$
2030   check.sum=VAL("&"+RIGHT$(postizo$,4))
2040   FOR puntero=1 TO LEN(postizo$)-5 STEP 2
2050     dummy=VAL("&"+MID$(postizo$,puntero,2))
2060     check.sum=check.sum-dummy
2070     direc=direc+1
2080     POKE direc,dummy
2090     NEXT puntero
2100   IF check.sum<>Ø GOTO 4Ø2Ø
2110   NEXT linea
2999 RETURN
4000 '
4020 PRINT CHR$(7)"Error en linea"linea
```

ANTA 64K.3

Los 64K de memoria que esperaba su Amstrad

*Ampliación de memoria, buffer de impresora y ram disk**



Si tiene un AMSTRAD CPC 464, CPC 664 o CPC 6128 conéctele el ANTA 64K.3 y seleccione la opción que necesite:

64K de Memoria

Para leer y escribir datos, cadenas y bloques de caracteres así como copiar o trasladar pantallas.

64K de Buffer de Impresora

Permite seguir trabajando con el ordenador mientras la impresora funciona.

64K de Ram Disk/Basic

La memoria simula el funcionamiento de un disco con mejor tiempo de acceso.

* *Software de manejo contenido en ROM*

MHT ingenieros

DISTRIBUIDO POR LSB, S. A. C/. SANCHEZ PACHECO, 78. 28002 MADRID. TEL. 413 92 68

Le esperamos en nuestros stands 9 y 10 de la 1.ª Feria Amstrad, desde el 23 al 25 de Mayo

SINCLAIR STORE

REGALO SEGURO



Por la compra de cualquier Ordenador el equipo completo oficial de Basket es tuyo.

- Como siempre curso gratis de informática.
- Somos distribuidores oficiales de todas las marcas.
- Teclado multifunción con sonido 13.200 ptas.
- Joystick+interface T. KEMPSTON 3.200 ptas.
- Lápiz óptico 3.500 ptas.
- Tarjeta de socio club Sinclair Store.

Además entre todos nuestros clientes sorteamos 10 lotes de 2 entradas para la final del MUNDIAL DE BASKET 86.



SOMOS PROFESIONALES

BRAVO MURILLO, 2
(Glorieta de Quevedo)
Tel. 446 62 31 - 28015 MADRID
Aparcamiento **GRATUITO** Magallanes, 1

DIEGO DE LEON, 25
(Esq. Nuñez de Balboa)
Tel. 261 88 01 - 28006 MADRID
Aparcamiento **GRATUITO** Nuñez de Balboa, 114

FELIPE II, 12
(Metro Goya)
Tel. 431 32 33 - 28 009 MADRID
Aparcamiento **GRATUITO** Felipe II